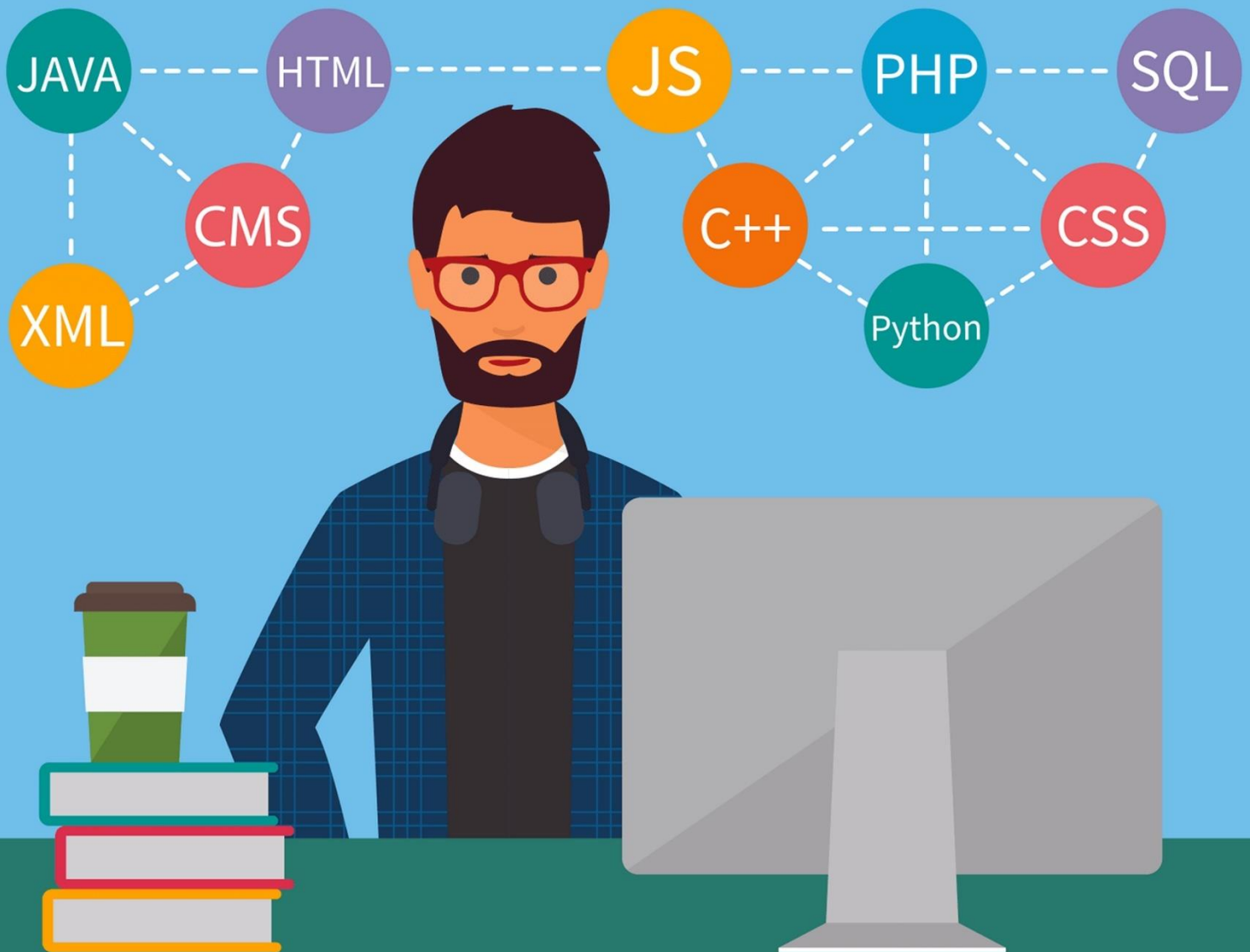


SUENDRI

MODUL PRAKTIKUM

Pemrograman Berbasis Web Lanjutan

Semester Genap 2019/2020



Digunakan oleh kalangan terbatas untuk keperluan matakuliah
Pemrograman Berbasis Web Lanjutan



Program Studi Sistem Informasi
Fakultas Sains dan Teknologi
Universitas Islam Negeri Sumatera Utara Medan

KATA PENGANTAR

Puji dan syukur kepada Allah 'Azza Wajalla yang telah melimpahkan rahmat dan kasih sayangNya sehingga Modul Praktikum ini selesai disusun dengan baik. Selawat dan salam juga teruntuk nabi junjungan alam, Muhammad Shollollohu 'Alaihi Wasallam sebagai panutan manusia hingga akhir zaman. Modul Praktikum ini disusun sebagai kelengkapan matakuliah Pemrograman Berbasis Web Lanjutan pada Program Studi Sistem Informasi, Universitas Islam Negeri (UIN) Sumatera Utara Medan. Modul Praktikum ini hanya untuk kepentingan perkuliahan, digunakan pada kalangan sendiri. Jika terdapat kutipan, seluruh referensi tersebut dicantumkan dengan lengkap pada daftar pustaka.

Penulis menyadari masih banyak perbaikan yang diperlukan dalam penyusunan, aturan penulisan dan tata letak dari Modul Praktikum ini, oleh karena itu penulis mengharapkan kritik dan saran yang membangun. Penulis juga mengucapkan terimakasih kepada seluruh pihak yang dijadikan referensi dalam penulisan Modul Praktikum ini, baik dari sisi perangkat lunak, perangkat keras serta buah pikiran dari pakar-pakar terdahulu.

Medan, Maret 2020

Suendri, M.Kom

DAFTAR ISI

KATA PENGANTAR	ii
DAFTAR ISI	iii
MODUL 1 : PHP OOP	1
1.1 Tujuan	1
1.2 Praktikum	1
1.3 Latihan	7
MODUL 2 : CLASS DAN OBJECT	8
2.1 Tujuan	8
2.2 Praktikum	8
2.3 Latihan	14
MODUL 3 : INHERITANCE	15
3.1 Tujuan	15
3.2 Praktikum	15
3.3 Latihan	24
MODUL 4 : ENCAPSULATION	25
4.1 Tujuan	25
4.2 Praktikum	25
4.3 Latihan	28
MODUL 5 : POLIMORPHISM	29
5.1 Tujuan	29
5.2 Praktikum	29
5.3 Latihan	35
MODUL 6 : NAMESPACE	36
6.1 Tujuan	36
6.2 Praktikum	36
6.3 Latihan	39
MODUL 7 : COMPOSER	40
7.1 Tujuan	40
7.2 Praktikum	40

7.3 Latihan	53
MODUL 8 : REPOSITORY	54
8.1 Tujuan	54
8.2 Praktikum	54
8.3 Latihan	66
MODUL 9 : MVC	67
9.1 Tujuan	67
9.2 Praktikum	67
9.3 Latihan	73
MODUL 10 : PHP FRAMEWORK	74
10.1 Tujuan	74
10.2 Praktikum	74
10.3 Latihan	80
10.4 Praktikum View	80
10.5 Latihan View	85
10.6 Praktikum Create	85
10.7 Latihan Create	88
10.8 Praktikum Edit	88
10.9 Latihan Edit	92

DAFTAR PUSTAKA

MODUL 1

PHP OOP

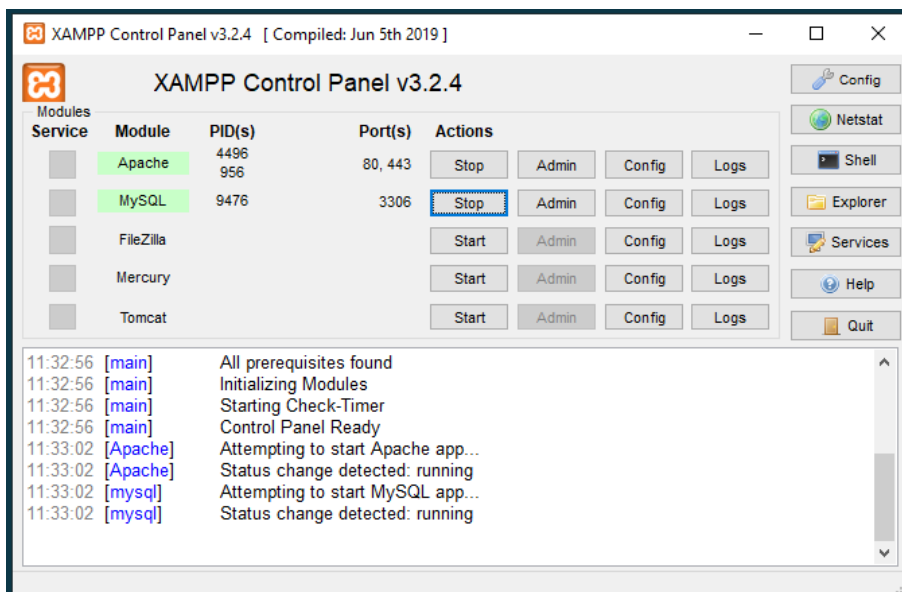
1.1 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

- Memahami dasar konsep OOP pada bahasa pemrograman PHP.
- Memahami penggunaan perangkat lunak yang digunakan dalam membangun konsep OOP pada bahasa pemrograman PHP.
- Merancang struktur *folder* dan *file* yang diperlukan dalam implementasi konsep OOP pada bahasa pemrograman PHP.

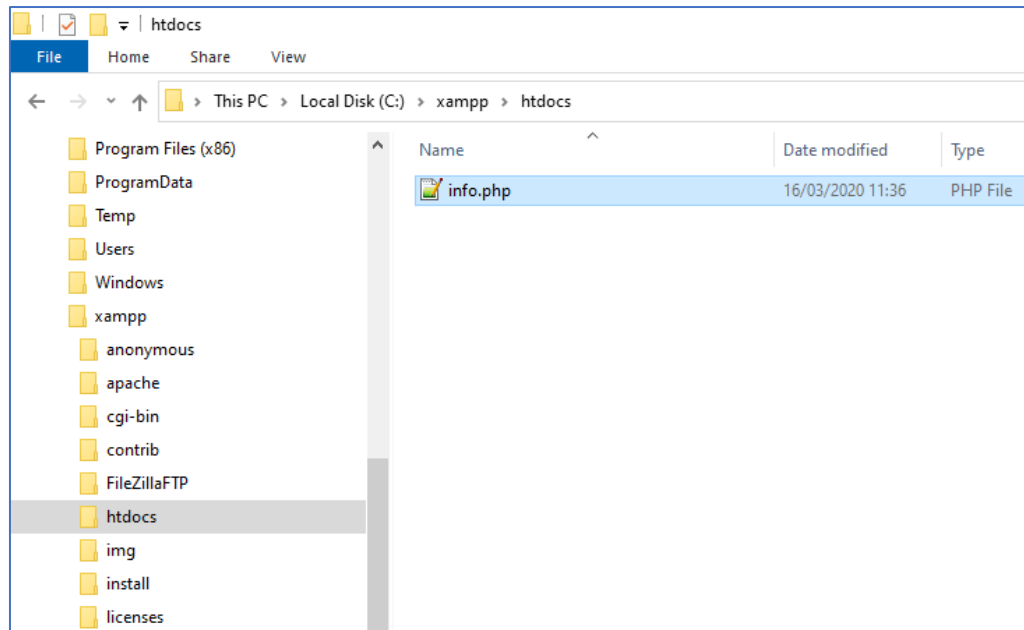
1.2 Praktikum

- Pastikan XAMPP 7.4.3 sudah terinstall dengan baik pada komputer masing-masing.
- Jalankan modul **Apache** dan **MySQL**.




- Buatlah sebuah *file* baru di **C:\xampp\htdocs** dengan nama **info.php**, buka *file* tersebut dan isikan perintah :

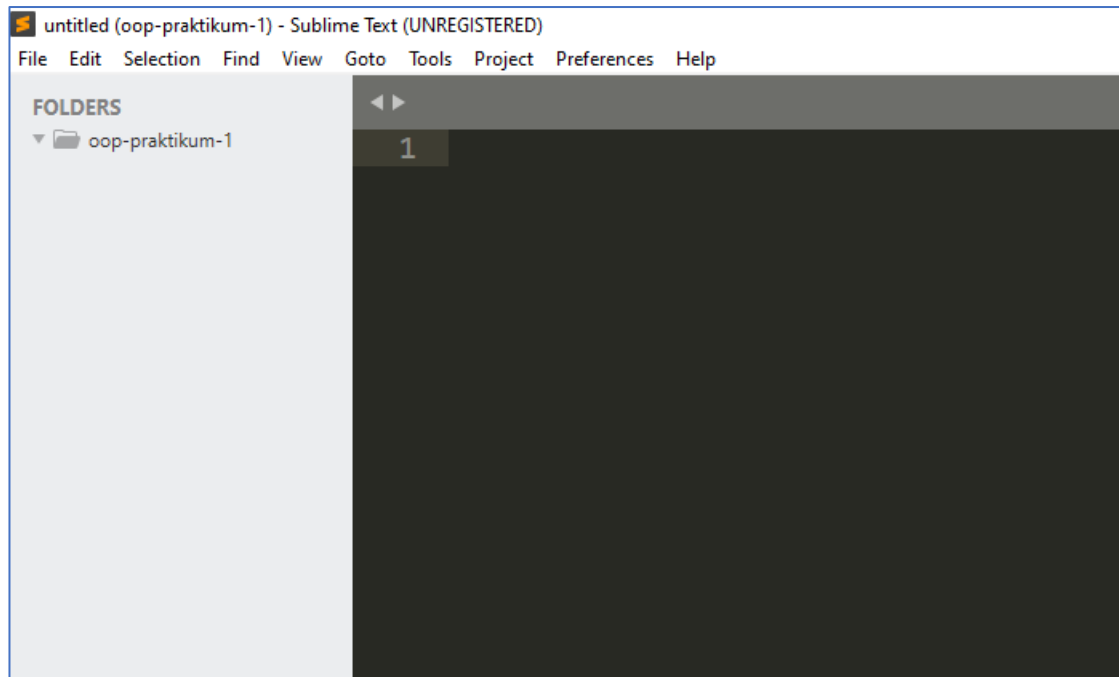
- `<?php`
-
- `phpinfo();`



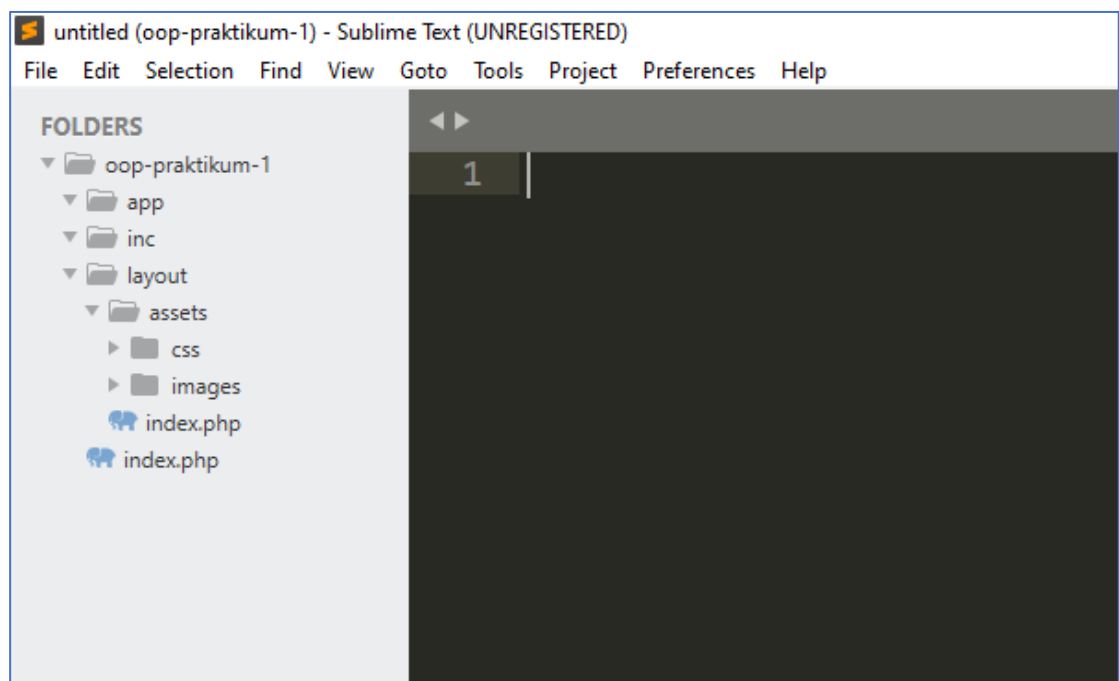
- d. *File info.php* ini berfungsi untuk menampilkan informasi PHP, pengaturan serta driver yang terpasang.

PHP Version 7.4.3	
	
System	Windows NT DESKTOP-F0CB97K 10.0 build 18363 (Windows 10) AMD64
Build Date	Feb 18 2020 17:23:22
Compiler	Visual C++ 2017
Architecture	x64
Configure Command	cmd /c "cd /d %~dp0\php\build\deps & call "C:\xampp\php\build\deps\php-builddeps_aux\oraclev64\instantclient_12_1\sdk\shared" --with-oci8-12c-c:\php-snap-build\deps_aux\oraclev64\instantclient_12_1\sdk\shared" --enable-object-out-dir=.\obj" --enable-com-dotnet=shared" --without-analyzer" --with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,TS,VC15
PHP Extension Build	API20190902,TS,VC15
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads

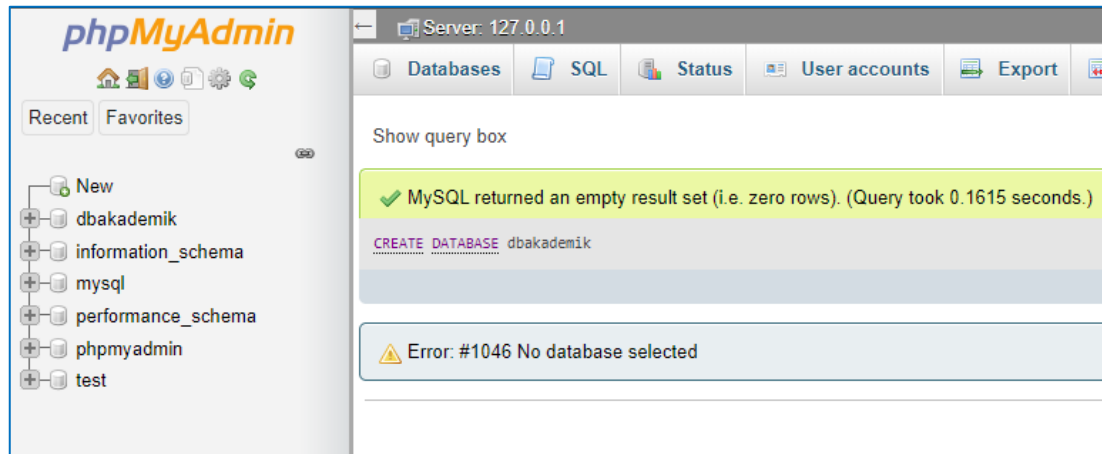
- e. Buatlah sebuah *folder* baru di **C:\xampp\htdocs** dengan nama ***oop-praktikum-1***. Bukalah text editor anda dan buka *folder* tersebut melalui text editor. Pada praktikum ini dan selanjutnya, kita menggunakan editor **Sublime Text 3**, namun anda tetap bisa menggunakan text editor yang menurut anda nyaman untuk digunakan. Jika anda belum memiliki Sublime Text 3, bisa langsung diunduh melalui web resminya <https://www.sublimetext.com/3>



- f. Buatlah *folder* dan *file* baru di dalam *folder* **oop-praktikum-1** dengan susunan seperti pada gambar berikut ini.

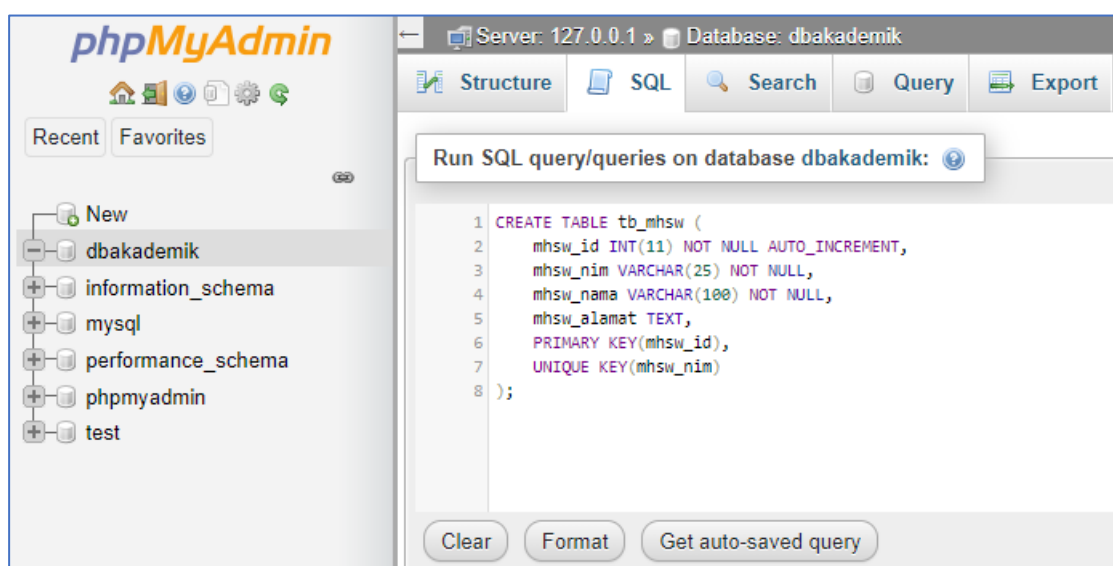


- g. Setelah *folder* kita siapkan, selanjutnya kita buat *database* menggunakan MySQL. Anda bisa menggunakan phpMyAdmin untuk pengaturan yang lebih mudah menggunakan *web browser*. Buatlah database baru dengan nama **dbakademik**.



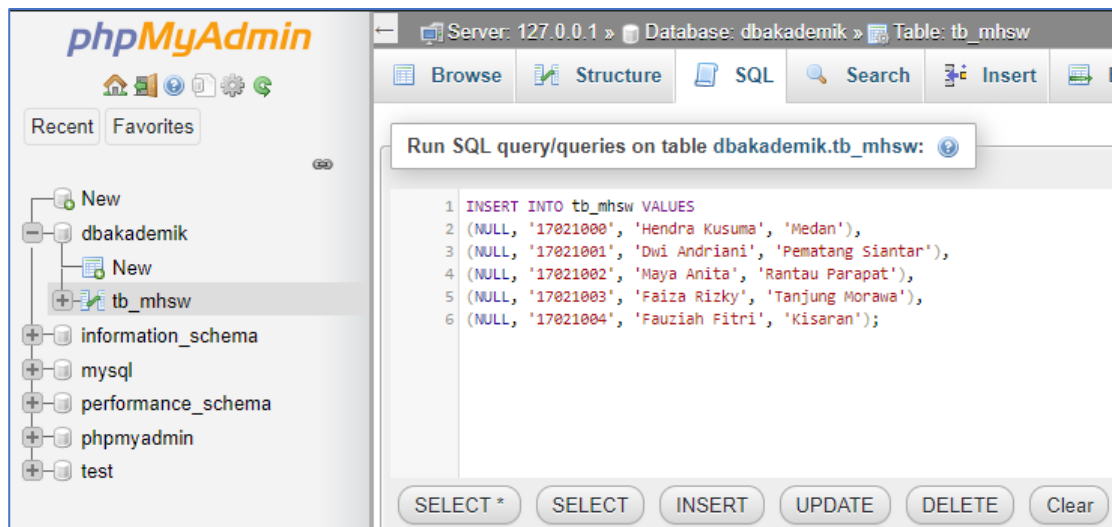
```
1. CREATE DATABASE dbakademik;
2.
3. CREATE TABLE tb_mhsw (
4.     mhsw_id INT(11) NOT NULL AUTO_INCREMENT,
5.     mhsw_nim VARCHAR(25) NOT NULL,
6.     mhsw_nama VARCHAR(100) NOT NULL,
7.     mhsw_alamat TEXT,
8.     PRIMARY KEY(mhsw_id),
9.     UNIQUE KEY(mhsw_nim)
10. );
```

- h. Kemudian buatlah sebuah tabel baru dengan nama **tb_mhsw**. Anda bisa menggunakan *query* diatas untuk tb_mhsw.



- i. Masukkan beberapa data berikut kedalam tabel **tb_mhsw**.

```
1. INSERT INTO tb_mhsw VALUES
2. (NULL, '17021000', 'Hendra Kusuma', 'Medan'),
3. (NULL, '17021001', 'Dwi Andriani', 'Pematang Siantar'),
4. (NULL, '17021002', 'Maya Anita', 'Rantau Parapat'),
5. (NULL, '17021003', 'Faiza Rizky', 'Tanjung Morawa'),
6. (NULL, '17021004', 'Fauziah Fitri', 'Kisaran');
```



- j. Buatlah *file* baru di dalam folder *app* dengan nama **Mhsw.php** dan isikan perintah berikut ini.

```
1. <?php
2.
3. class Mhsw {
4.
5.     private $db;
6.
7.     public function __construct()
8.     {
9.         try {
10.
11.             $this->db =
12.             new PDO("mysql:host=localhost;dbname=dbakademik", "root", "");
13.
14.         } catch (PDOException $e) {
15.             die ("Error " . $e->getMessage());
16.         }
17.     }
18.
19.     public function tampil()
20.     {
21.         $sql = "SELECT * FROM tb_mhsw";
```

```
22.         $stmt = $this->db->prepare($sql);
23.         $stmt->execute();
24.
25.         $data = [];
26.         while ($rows = $stmt->fetch()) {
27.             $data[] = $rows;
28.         }
29.
30.         return $data;
31.     }
32. }
```

- k. Selanjutnya file **index.php** pada *root folder* sempurnakan dengan menambahkan perintah berikut ini.

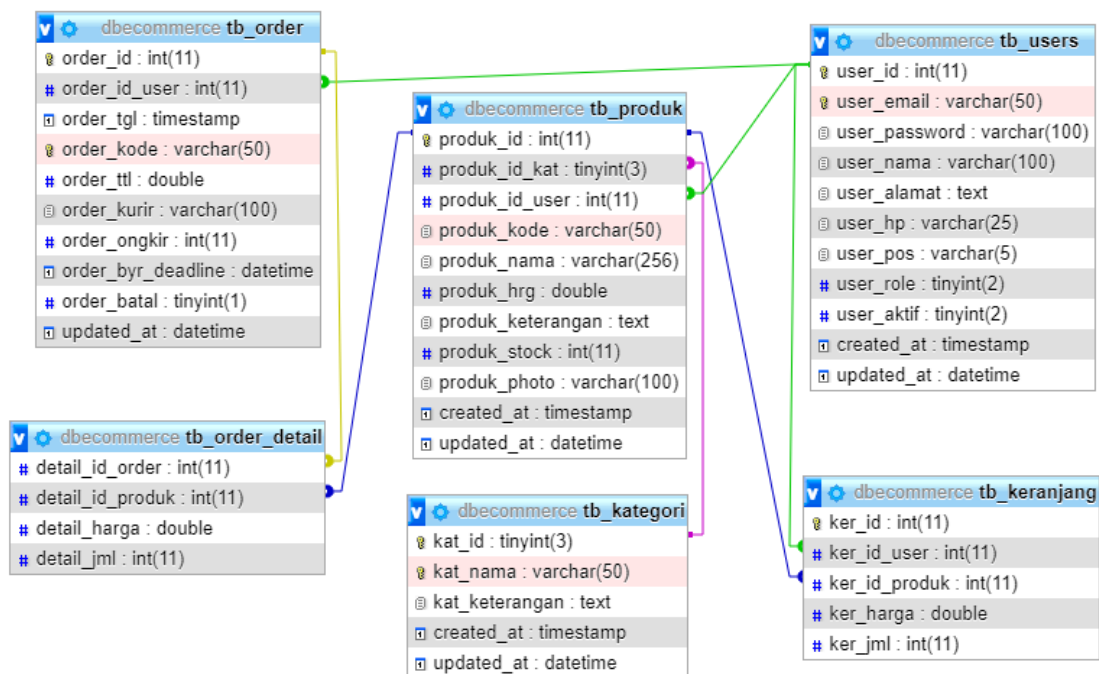
```
1. <?php
2.
3. require_once "app/Mhsw.php";
4.
5. $mhsw = new Mhsw();
6. $rows = $mhsw->tampil();
7.
8. ?>
9.
10. <table>
11.     <tr>
12.         <td>NO</td>
13.         <td>NIM</td>
14.         <td>NAMA</td>
15.         <td>ALAMAT</td>
16.     </tr>
17.
18.     <?php foreach ($rows as $row) { ?>
19.
20.         <tr>
21.             <td><?php echo $row['mhsw_id']; ?></td>
22.             <td><?php echo $row['mhsw_nim']; ?></td>
23.             <td><?php echo $row['mhsw_nama']; ?></td>
24.             <td><?php echo $row['mhsw_alamat']; ?></td>
25.         </tr>
26.
27.     <?php } ?>
28. </table>
```

- l. Jalankan sistem yang anda buat dengan membuka alamat <http://localhost/oop-praktikum-1/> pada URL Browser anda.

Oke, untuk sesi ini sudah selesai, berikutnya bagian anda. Selesaikan Latihan dibawah menggunakan langkah-langkah yang telah kita bahas pada halaman sebelumnya.

1.3 Latihan

1. Jelaskanlah bagian dari program **oop-praktikum-1** yang menjadi ciri khas konsep OOP.
2. Rancanglah database menggunakan MySQL dengan desain seperti pada gambar berikut ini, kemudian bangunlah sistem sederhana dengan konsep OOP sesuai dengan praktikum sebelumnya.



MODUL 2

CLASS DAN OBJECT

2.1 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

- Memahami konsep *class* dan langkah pembuatan *class* menggunakan bahasa pemrograman PHP.
- Memahami dan mengetahui bagian dan fungsi yang membangun sebuah *class*.
- Memahami dan merancang sistem yang dilengkapi dengan *class* beserta bagian dan fungsi yang terdapat di dalam *class* tersebut dengan baik.

2.2 Praktikum

- Class*

Untuk membuat sebuah *class*, perintah yang digunakan adalah *class* yang diikuti dengan nama *class* yang diinginkan. Baca kembali diktat teori untuk mengetahui cara pembuatan nama kelas yang baik. Nama *file* yang menyimpan *class* ini sama dengan nama *class*.

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     // class
6.
7. }
```

Buatlah sebuah *file* dengan nama *Mahasiswa.php* dan masukkan perintah diatas untuk membuat sebuah *class* baru. Jika *file* tersebut diakses melalui *localhost*, maka tampilan *file* hanya kosong saja, karena belum ada *object* yang diciptakan dari *class* tersebut.

b. Property

Property merupakan variabel yang terletak di dalam *class*, masing-masing *property* mempunyai *access modifier*. *Access modifier* merupakan cara bagaimana hak akses terhadap variabel tersebut, akses tersebut yaitu : (1) **Public**, (2) **Private**, (3) **Protected**.

Dengar penjelasan dosen pengampu untuk penjelasan *Property* ini, buatlah sebuah *file* baru dan masukkan perintah berikut ini:

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     // Access Modifier, Typed Properties dan Property
6.
7.     // Bisa diakses class yang sama dan class lain.
8.     public string $nim;
9.     public string $nama;
10.    public int $umur;
11.
12.    // Hanya bisa diakses dalam class yang sama.
13.    private string $email;
14.
15.    // Hanya bisa diakses dalam class yang sama dan anak-anaknya.
16.    protected string $nama_ibu;
17.
18.    /* 1. bool
19.        2. int
20.        3. float
21.        4. string
22.        5. array
23.        6. iterable
24.        7. object
25.        8. ?(nullable)
26.        9. self & parent
27.        10. Classes & interfaces
28.    */
29.
30. }
```

c. Method

Method merupakan *function* yang terdapat di dalam *class*, *method* juga memiliki *Access modifier* sama dengan *property*. *Method* setara dengan *property*, jika terdapat variabel di dalam *method*, maka itu tidak disebut dengan *property*, tapi hanya *local* variabel saja. Perhatikan perintah beriku ini, buatlah *file* baru dan masukkan perintah tersebut.

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     // Method
6.
7.     public function setNim(string $nim) {
8.         return $nim;
9.     }
10.
11.    public function setName(string $nama) {
12.        return $nama;
13.    }
14.
15.    public function setUmur(int $umur) {
16.        return $umur;
17.    }
18. }
```

d. Constructor

Constructor merupakan *method default* yang dipanggil oleh *class* ketika tidak disebutkan *method* khusus pada *object* dari *class* tersebut. Perintah yang digunakan untuk *constructor* ini adalah :

```
__constructor($var, $var, ...)
```

Buatlah sebuah *file* baru dan masukkan perintah beriku ini:

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     public string $nim;
6.     public string $nama;
7.
8.     // Constructor
9.
10.    public function __construct(string $a, string $b) {
11.
12.        $this->nim = $a;
13.        $this->nama = $b;
14.    }
15. }
```

e. *Instantiation*

Instantiation atau *Instance* merupakan proses utama yang menjadi *class* ke dalam *object*. Proses ini yang menjadi ujung dari konsep OOP pada bahasa pemrograman PHP. Seluruh *Property* dan *Method* yang ada dalam *class* digunakan oleh *object* sesuai dengan tujuan dan *access modifier*-nya. Proses *Instance* menggunakan perintah “**new**”. Buatlah *file* baru dan bahaslah kode perintah berikut ini dengan seksama.

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     public string $nim;
6.     public string $nama;
7.     public static string $agama = "Islam";
8.
9.     public function setNim(string $nim) {
10.         return $nim;
11.     }
12.
13.     public function setNama(string $b) {
14.
15.         // $this keyword refers a non-
16.         static member of a class
17.         return $this->nama = $b;
18.     }
19.     public function getNama() {
20.
21.         // $this keyword refers a non-
22.         static member of a class
23.         return $this->nama;
24.     }
25.     public static function getAgama() {
26.
27.         // self keyword refers a static member of a class
28.         return self::$agama;
29.     }
30. }
31.
```

```
32. // Instantiation
33.
34. $mhs = new Mahasiswa();
35.
36. echo $mhs->setNim('17021000');
37.
38. $mhs->setNama('Faiza');
39.
40. echo $mhs->getNama();
41.
42. echo $mhs->getAgama();
```

perintah **new** pada baris 34 menjadikan \$mhs sebagai *object* dari *class* Mahasiswa()

f. Static Keyword

Static keyword digunakan pada *Property* dan *Method* berfungsi agar bisa dipanggil langsung pada objek tanpa melalui proses *instantiation*.

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     public static string $agama = "Islam";
6.
7.     public static function getAgama() {
8.
9.         //
10.
11.     }
12. }
```

g. Scope Resolution Operator

Scope Resolution Operator atau double colon (::) atau titik dua, dua kali, merupakan tanda yang digunakan pada proses *Instantiation* dari *Property* atau *Method Static* (*Static keyword*). Lihat materi pada huruf f untuk *Static keyword*. Buatlah sebuah *file* baru dan coba perintah dibawah untuk *Scope Resolution Operator*


```
1. <?php
2.
3. class Mahasiswa {
4.
5.     public static function setNama(string $nama) {
6.
7.         return $nama;
8.
9.     }
10. }
11.
12. // Instantiation with Scope resolution operator
13. // Paamayim Nekudotayim
14.
15. echo Mahasiswa::setNama('Faiza');
```

h. Error Handling

Error Handling berfungsi untuk menangani *error* yang terdapat di dalam *class*. Untuk penjelasan lebih lanjut, silakan baca kembali Diktat perkuliahan pada sesi teori. Buatlah sebuah *file* baru dan bahas kode perintah berikut ini.

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     public int $umur = 22;
6.
7.     public function getUmur() {
8.
9.         try {
10.
11.             if ($this->umur < 25) {
12.                 throw new Exception('Anda masih muda');
13.             }
14.
15.         } catch (Exception $e) {
16.
17.             die ("Maaf Error, " . $e->getMessage());
18.
19.         }
20.     }
21. }
22.
23. $mhs = new Mahasiswa();
24. $mhs->getUmur();
```

2.3 Latihan

- a. Sempurnakanlah Latihan pada praktikum 1 dengan mengimplementasikan bagian dan fungsi *class* sesuai dengan materi praktikum 2.
- b. Lengkapi sistem pada latihan praktikum 2 dengan perintah input, edit dan delete data sesuai dengan database praktikum 1.

MODUL 3

INHERITANCE

3.1 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

- Memahami metode *Inheritance* (Pewarisan) dalam konsep OOP pada bahasa pemrograman PHP.
- Memahami dan mengetahui bagian dan fungsi yang membangun sebuah anak *class* dari *class* lainnya.
- Memahami dan merancang sistem yang dilengkapi dengan anak *class* beserta bagian dan fungsi yang terdapat di dalam anak *class* tersebut dengan baik.

3.2 Praktikum

- Anak *class*

Anak *class* merupakan *class* turunan dari sebuah *class* yang dapat mewarisi atau menggunakan seluruh *resource* yang diizinkan oleh *class* yang menjadi induknya. Dengan konsep pewarisan ini, maka *property* dan *method* yang ada pada *class* induk tidak lagi perlu ditulis ulang pada *class* anak, kecuali dalam beberapa kasus tertentu yang dibutuhkan. Perintah yang digunakan adalah “**extends**”. Perhatikan kode perintah berikut ini, perhatikan dengan seksama dan bahaslah bersama dosen pengampu.

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     public string $nim;
6.     public string $nama;
7.     public int $umur;
8.     private string $email;
9.     protected string $nama_ibu;
10.
11.     public function setNim(string $a) {
12.         $this->nim = $a;
13.     }
14.
15.     public function setNama(string $b) {
```

```
16.         $this->nama = $b;
17.     }
18.
19. }
20.
21. class Nilai extends Mahasiswa {
22.
23.     public function getNim() {
24.         return $this->nim;
25.     }
26.
27.     public function getNama() {
28.         return $this->nama;
29.     }
30.
31.     public function setIbu(string $c) {
32.         $this->nama_ibu = $c;
33.     }
34.
35.     public function getIbu() {
36.         return $this->nama_ibu;
37.     }
38. }
39.
40. $nilai = new Nilai();
41. $nilai->setNim("17021000");
42. $nilai->setNama("Faiza");
43. $nilai->setIbu("Fifi");
44.
45. echo $nilai->getNim() . " " . $nilai->getNama()
46. . " " . $nilai->getIbu();
```

Perhatikan baris ke 21 pada kode perintah diatas.

```
class Nilai extends Mahasiswa {
```

Class **Nilai** merupakan Anak *class* dari *class* **Mahasiswa** yang menjadi *class* utama. Seluruh *property* dan *method* dari *class* **Mahasiswa** bisa digunakan oleh *class* **Nilai**. *Class* **Nilai** juga bisa membuat *property* dan *method* sendiri.

b. Anak dari Anak *Class*

Class yang telah menjadi anak *class* dari *class* sebelumnya, bisa juga menjadi *class* induk bagi *class* lainnya. Perhatikan kode perintah berikut ini yang merupakan

kasus yang sama dengan kode perintah sebelumnya. Namun *class* tersebut menjadi anak dari anak *class* lainnya.

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     public string $nim;
6.     public string $nama;
7.
8.     public function setNim(string $a) {
9.         $this->nim = $a;
10.    }
11.
12.    public function setNama(string $b) {
13.        $this->nama = $b;
14.    }
15. }
16.
17. class Krs extends Mahasiswa {
18.
19.     protected string $matakuliah;
20.
21.     public function getNim() {
22.
23.         return $this->nim;
24.     }
25.
26.     public function getNama() {
27.
28.         return $this->nama;
29.     }
30.
31.     public function setMatakuliah(string $c) {
32.         $this->matakuliah = $c;
33.     }
34. }
35.
36. class Nilai extends Krs {
37.
38.     private static int $nilai = 90;
39.
40.     public function getMatakuliah() {
41.
42.         return $this->matakuliah;
43.     }
44. }
```

```
45.     public static function getNilai() {
46.
47.         return self::$nilai;
48.     }
49. }
50.
51. $nilai = new Nilai();
52.
53. // Memanggil class Mahasiswa
54. $nilai->setNim("17021000");
55. $nilai->setNama("Faiza");
56.
57. // Memanggil class Krs
58. $nilai->
59. setMatakuliah("Pemrograman Berbasis Web Lanjutan");
60.
61. echo "<p>NIM = " . $nilai->getNim() . "</p>";
62. echo "<p>Nama = " . $nilai->getNama() . "</p>";
63. echo "<p>Matakuliah = " . $nilai->
64. getMatakuliah() . "</p>";
65. echo "<p>Nilai = " . Nilai::getNilai(90) . "</p>";
```

Pada kode perintah diatas terdapat tiga *class* yaitu: **Mahasiswa**, **Krs** dan **Nilai**. Pada baris 17, *class Krs* menjadi pewaris dari *class Mahasiswa*, sedangkan pada baris 36 *class Nilai* menjadi Pewaris dari *class Krs*.

c. Tamu Class

Tamu *class* yang dimaksud pada pembahasan ini adalah *class* yang di-*instance* kedalam *class* lainnya. Banyak kasus yang menggabungkan beberapa *class* kedalam *class* lainya, namun *class* tersebut bukan merupakan anak dari *class* lainnya tersebut. Perhatikan contoh berikut ini, buat *file* baru dan implementasikan kode perintah berikut .

```
1. <?php
2.
3. class Fakultas {
4.
5.     public string $nama;
6.
7.     public function __construct(string $a) {
8.         $this->nama = $a;
9.     }
10. }
```

```
11.
12. class Prodi {
13.
14.     public static function setNama(string $b) {
15.         return $b;
16.     }
17. }
18.
19. class Mahasiswa {
20.
21.     public static function getFakultas() {
22.
23.         $fak = new Fakultas("FST");
24.         return $fak->nama;
25.     }
26.
27.     public static function getProdi() {
28.
29.         return Prodi::setNama("Sistem Informasi");
30.     }
31. }
32.
33. // $fak = new Fakultas("FST");
34. // echo $fak->nama;
35.
36. // echo Prodi::setNama("Sistem Informasi");
37.
38. echo Mahasiswa::getFakultas() . " "
39. . Mahasiswa::getProdi();
```

Pada kode perintah tersebut diatas terdapat tiga *class* yaitu: **Fakultas**, **Prodi** dan **Mahasiswa**. Masing-masing *class* berdiri sendiri dan terpisah. Didalam *class Mahasiswa* terdapat *class Fakultas* dan **Prodi**, perhatikan baris 23 dan baris 29. *Class Fakultas* dijadikan *Object* *\$fak*, sedangkan *class Prodi* langsung menggunakan *Scope Resolution Operator* atau double colon (::) karena mempunyai *method static*.

d. Protected Modifier

Protected Modifier merupakan *modifier* yang hanya membolehkan akses terhadap *property* atau *method* dari dalam *class* itu sendiri atau dari anak *class*-nya. Silakan lihat pembahasan bagian pertama dari praktikum ini. Perhatikan baris perintah

```
protected string $nama_ibu;
```

`$nama_ibu` hanya bisa diakses oleh *class* itu sendiri atau anak *class*-nya.

e. *Overriding*

Overriding merupakan sebuah *method* yang terdapat pada anak *class* yang menempa *method* induknya. Nama *method* ini sama dengan *method* induk, sehingga *method* yang dipanggil adalah *method* anak saja, sedangkan *method* induknya diabaikan. Perhatikan contoh kode perintah berikut ini.

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     public string $nim = "17021000";
6.
7.     public function setNim() {
8.
9.         return $this->nim;
10.    }
11. }
12.
13. class Krs extends Mahasiswa {
14.
15.     // Method override
16.
17.     public function setNim() {
18.
19.         return null;
20.    }
21. }
22.
23.
24. $krs = new Krs();
25. echo $krs->setNim();
```

Perhatikan baris 7 dan baris 17, *method* tersebut mempunyai nama yang sama, setelah dieksekusi pada *class* anak, maka *method* anak menempa hasil dari *method* induknya.

f. *Self* dan *Parent*

Self digunakan untuk mengacu terhadap bagian dan fungsi *class* itu sendiri, sedangkan *Parent* digunakan untuk mengacu terhadap bagian dan fungsi *class* induknya. Perhatikan kode perintah berikut ini, buatlah *file* baru dan implementasikan kode perintah tersebut serta lakukan pembahasan bersama dosen pengampu.

```
1. <?php
2.
3. class Prodi {
4.
5.     public static string $nama = "Sistem Informasi";
6.
7.     public static function setNama() {
8.         return self::$nama;
9.     }
10. }
11.
12. class Mahasiswa extends Prodi {
13.
14.     public static function setProdi() {
15.
16.         return parent::setNama();
17.     }
18. }
19.
20. echo Mahasiswa::setProdi();
```

Perhatikan baris 8 dan baris 16, terdapat perintah *self* yang mengacu kepada *class Prodi* sedangkan *parent* pada *class Mahasiswa* kembali mengacukan kepada *class* induknya tersebut yaitu **Prodi**.

g. *Final Keyword*

Sesuai dengan namanya, perintah *final* ini untuk menyatakan bahwa *method* tidak bisa dilagi diubah oleh *class* lain yang menjadi anak *class*. *Final* bisa digunakan pada *method* atau pada *anak class* paling bawah, *method* dengan *keyword final* berisi nilai permanen dan tidak bisa lagi ditimpa. Perhatikan contoh kode perintah berikut ini:

```
1. <?php
2.
3. class Prodi {
4.
5.     public static string $nama = "Sistem Informasi";
6.
7.     final static function setNama() {
8.         return self::$nama;
9.     }
10. }
11.
12. final class Mahasiswa extends Prodi {
13.
14.     public static function setProdi() {
15.
16.         return parent::setNama();
17.     }
18. }
19.
20. echo Mahasiswa::setProdi();
```

Perhatikan pada baris 7 dan baris 12. *Keyword final* pada baris 7 digunakan pada *method class Prodi*, *method* ini bersifat *final* dan tidak bisa ditimpa oleh *class Mahasiswa* sebagai anak *class*. *Keyword final* pada baris 12 digunakan pada *class Mahasiswa*, *class* ini bersifat *final* dan tidak bisa lagi mempunyai anak *class* dibawahnya.

h. Trait

Bagaimana jika sebuah *class* anak ingin mewarisi lebih dari 1 *class* induk? Tentu tidak diizinkan dalam bahasa pemrograman PHP. Satu *class* anak hanya bisa mewarisi satu *class* induk. Nah, sebagai jalan keluarnya, PHP menyediakan sebuah perintah yang mirip dengan *class* namun tidak bisa dijadikan *object*. Perintah *Trait* bisa digabungkan beberapa *Trait* dan bisa mewarisi bagian dan fungsi dari dalam *Trait* tersebut kepada *class* anak. Perhatikan kode perintah berikut ini, bahaslah kode perintah tersebut bersama dosen pengampu. Buat studi kasus yang berbeda agar lebih mudah diingat.

```
1. <?php
2.
3. trait Fakultas {
4.
5.     // Boleh juga ada property
6.
7.     public function setFakultas(string $a) {
8.         return $a;
9.     }
10. }
11.
12. trait Prodi {
13.
14.     // Boleh juga ada property
15.
16.     public function setProdi(string $b) {
17.         return $b;
18.     }
19. }
20.
21. class Mahasiswa {
22.
23.     use Fakultas, Prodi;
24.     public string $nama;
25.
26.     public function setName(string $c) {
27.
28.         $this->nama = $c;
29.     }
30. }
31.
32. $mhs = new Mahasiswa();
33. $mhs->setName("Faiza");
34. echo $mhs->setFakultas("FST") . " "
35. . $mhs->setProdi("Sistem Informasi") . " "
36. . $mhs->nama;
```

Perhatikan baris 3 dan 12, perintah tersebut merupakan perintah *trait* yang mirip dengan *class*, *trait* juga bisa memiliki *property* dan *method*, namun tidak bisa dijadikan *object*. Pada baris 23, *class Mahasiswa* menggunakan *trait Fakultas* dan *Prodi*. *Property* dan *method* dari *trait* tersebut bisa digunakan oleh *class mahasiswa*.

3.3 Latihan

- a. Sempurnakanlah Latihan pada praktikum 2 dengan mengimplementasikan metode pewarisan sesuai dengan materi praktikum 3.
- b. Dosen pengampu menyediakan video implementasi Praktikum 1 - 3, simak videonya pada link yang disediakan, bandingkan dengan program yang telah anda rancang, masing-masing mahasiswa wajib berikan ulasan dan pendapat anda pada video yang disediakan.
- c. Jawablah quiz yang disediakan di *e-learning* pada pertemuan 3 untuk menguji kemampuan anda sampai dengan materi praktikum ini.

MODUL 4

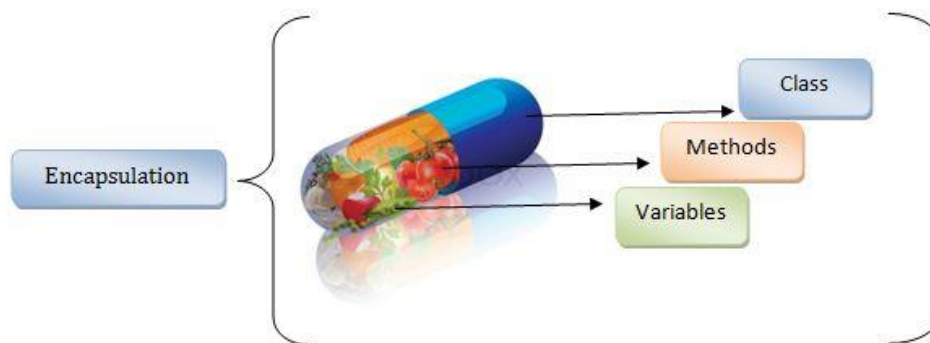
ENCAPSULATION

4.1 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

- Memahami metode *Encapsulation* (Pembungkusan) dalam konsep OOP pada bahasa pemrograman PHP.
- Memahami dan mengetahui bagian dan fungsi yang membangun metode *encapsulation* pada bahasa pemrograman PHP.
- Memahami dan merancang sistem yang dilengkapi dengan metode *encapsulation*.

4.2 Praktikum



Encapsulation (Pembungkusan) sebenarnya bukanlah proses yang baru dari materi yang sudah kita bahas pada praktikum sebelumnya. *Encapsulation* adalah proses membungkus *class* dan menjaga apa saja yang ada di dalam *class* tersebut agar tidak bisa diakses sembarangan dari *class* lainnya. Perintah yang sudah kita bahas yaitu penggunaan *Modifier Private* dan *Protected* pada *Property* maupun *Method*. Dengan *Modifier Private* maka *Property* dan *Method* hanya bisa diakses oleh *class* itu sendiri, sedangkan *Modifier Protected* hanya bisa diakses oleh *Class* itu sendiri dan Anak *Class*-nya. Berikut ini keuntungan dari penggunaan *encapsulation* pada konsep OOP menggunakan bahasa pemrograman PHP

- Penyembunyian data, struktur dari data dan detail yang tidak perlu diketahui pengguna dapat disembunyikan.

- b. Keamanan data, proses encapsulation membuat data lebih kuat dan aman, hal ini disebabkan data terlindungi dan tidak ada pengaruh dari luar.
- c. Mengurangi kompleksitas karena data tersembunyi.
- d. Dapat digunakan kembali, property dan method dengan telah dimiliki oleh Class bisa digunakan kembali oleh Anak Class tanpa perlu ditulis ulang.
- e. Handal, data bisa dijadikan hanya bisa di baca (*read-only*) atau hanya ditulis (*write-only*).
- f. Mudah untuk diuji, penggunaan method pada anak class, memastikan method pada class induk sudah benar.
- g. Meningkatkan fleksibilitas, bisa diimplementasikan pada class lain tanpa merubah kode.

a. *Private*

Private hanya bisa diakses *class* itu sendiri, berikut contoh penggunaan *private*:

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     private string $nim;
6.     public string $nama;
7.
8.     public function setNim(string $a) {
9.         $this->nim = $a;
10.    }
11.
12.    public function setNama(string $nama) {
13.        $this->nama = $nama;
14.    }
15.
16.    public function getNim() {
17.        return $this->nim;
18.    }
19.
20. }
21.
22. $mhs = new Mahasiswa;
23. $mhs->setNim("17021000");
24. $mhs->setNama("Kiki");
25.
26. echo "<p>" . $mhs->getNim() . "</p>";
27. echo "<p>" . $mhs->nama . "</p>";
```

Pada contoh di atas, `$mhs->nama` bisa diakses langsung pada saat ditampilkan ke layar, tapi tidak sebaliknya dengan `$mhs->nim` karena *modifier*-nya *private*, maka dibantu dengan method `$mhs->getNim()`.

b. Protected

Protected bisa diakses oleh *class* itu sendiri dan turunannya, berikut ini contoh penggunaan *protected*:

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     private string $nim = "17021000";
6.     protected string $nama = "Kiki";
7.
8. }
9.
10. class Nilai extends Mahasiswa {
11.
12.     public function getNim() {
13.         return $this->nim;
14.     }
15.
16.     public function getNama() {
17.         return $this->nama;
18.     }
19. }
20.
21. $mhs = new Nilai();
22.
23. echo "<p>" . $mhs->getNim() . "</p>";
24. echo "<p>" . $mhs->getNama() . "</p>";
```

Pada contoh diatas `$mhs->getNim()` tidak bisa dieksekusi karena `$nim` mempunyai *modifier private*, sedangkan `$mhs->getNama()` bisa dieksekusi karena mempunyai *modifier protected*.

Pada dua contoh diatas digunakan pada *Property*, begitu juga penggunaan pada *Method* dengan cara yang sama.

4.3 Latihan

- a. Jawablah quiz yang disediakan di *e-learning* pada pertemuan 4 untuk menguji kemampuan anda sampai dengan materi praktikum ini.

MODUL 5

POLYMORPHISM

5.1 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

- Memahami metode *Polymorphism* (Banyak bentuk) dalam konsep OOP pada bahasa pemrograman PHP.
- Memahami dan mengetahui bagian dan fungsi yang membangun metode *Polymorphism* pada bahasa pemrograman PHP.
- Memahami dan merancang sistem yang dilengkapi dengan metode *Polymorphism*.

5.2 Praktikum

Dari segi bahasa, Polimorfisme (bahasa inggris: *Polymorphism*) berasal dari dua kata bahasa latin yakni *poly* dan *morph*. *Poly* berarti banyak, dan *morph* berarti bentuk. Polimorfisme berarti banyak bentuk. *Polymorphism* merupakan konsep dimana terdapat banyak *class* yang memiliki *signature method* yang sama. *Signature method* adalah nama *method* tersebut beserta parameternya jika ada. Implementasi dari *method-method* tersebut diserahkan kepada tiap *class*, akan tetapi cara pemanggilan *method* harus sama. Agar kita dapat 'memaksakan' *signature method* yang sama pada banyak *class*, *class* tersebut harus diturunkan dari sebuah *abstract class* atau *object interface* (Dunia Ilkom). Untuk lebih dipahami, kita akan bagi praktikum ini dalam beberapa sesi.

a. Abstract Class

Sesuai dengan namanya, *abstract class* bukan kelas yang sebenarnya, *class* ini hanya menjadi kerangka dasar bagi Anak *Class*-nya. *Abstract class* tidak bisa di Instansiasi karena bukan *class* sebenarnya. Di dalam *Abstract Class* juga terdapat *Abstract method*, *Abstract Method* adalah *method* dasar yang harus diimplementasikan ulang di dalam *class* anak. *Abstract method* ditulis tanpa isi dari *method*, melainkan hanya *signature*-nya saja. Untuk lebih dipahami, perhatikan contoh berikut ini:

```
1. <?php
2.
3. abstract class Peserta {
4.
5.     abstract protected function dataDiri();
6.     abstract protected function dataOrtu();
7.     abstract protected function dataSekolahAsal();
8.
9. }
10.
11. class Siswa extends Peserta {
12.
13.     public string $nama = "Kiki";
14.
15.     public function dataDiri() {
16.         return $this->nama;
17.     }
18.
19.     public function dataOrtu() {
20.         //
21.     }
22.
23.     public function dataSekolahAsal() {
24.         //
25.     }
26.
27. }
28.
29. class Mahasiswa extends Peserta {
30.
31.     public string $nama = "Zizi";
32.
33.     public function dataDiri() {
34.         return $this->nama;
35.     }
36.
37.     public function dataOrtu() {
38.         //
39.     }
40.
41.     public function dataSekolahAsal() {
42.         //
43.     }
44. }
45.
46. $ssw = new Siswa;
47. echo "<p>" . $ssw->dataDiri() . "</p>";
48.
49. $mhs = new Mahasiswa;
50. echo "<p>" . $mhs->dataDiri() . "</p>";
```

Pada contoh diatas, Peserta merupakan sebuah *Abstract class* yang merupakan kerangka dasar untuk semua Anak *Class*-nya.

b. Interface

Interface atau *Object Interface* pada dasarnya hampir sama dengan *Abstract class*, namun pastinya tujuan dan cara penggunaannya berbeda. *Interface* ini hanya menjadi kerangka dasar bagi *class* lain yang mengimplmentasikannya. Kerangka yang telah dibuat *Interface* tidak bisa di Instansiasi karena tidak sama dengan *class* sebenarnya. Di dalam *Interface* juga terdapat *method* dasar yang harus diimplementasikan ulang di dalam *class* yang mengimplementasikannya. *Method* tersebut ditulis tanpa isi dari *method*, melainkan hanya *signature*-nya saja. Untuk lebih dipahami, perhatikan contoh berikut ini:

```
1. <?php
2.
3. interface Peserta {
4.
5.     public function dataDiri();
6.     public function dataOrtu();
7.     public function dataSekolahAsal();
8.
9. }
10.
11. class Siswa implements Peserta {
12.
13.     public string $nama = "Kiki";
14.
15.     public function dataDiri() {
16.         return $this->nama;
17.     }
18.
19.     public function dataOrtu() {
20.         //
21.     }
22.
23.     public function dataSekolahAsal() {
24.         //
25.     }
26.
27. }
28.
29. class Mahasiswa implements Peserta {
30.
31.     public string $nama = "Zizi";
32.
33.     public function dataDiri() {
34.         return $this->nama;
35.     }
36.
```

```
37.     public function dataOrtu() {
38.         //
39.     }
40.
41.     public function dataSekolahAsal() {
42.         //
43.     }
44. }
45.
46. $ssw = new Siswa;
47. echo "<p>" . $ssw->dataDiri() . "</p>";
48.
49. $mhs = new Mahasiswa;
50. echo "<p>" . $mhs->dataDiri() . "</p>";
```

Interface bisa mempunyai Anak Interface

Sebuah *interface* bisa diturunkan ke *interface* lain selayaknya *class* dengan perintah *extends*.

```
1. <?php
2.
3. interface Formasi {
4.     public function dataFormasi();
5. }
6.
7. interface Peserta extends Formasi {
8.
9.     public function dataDiri();
10.    public function dataOrtu();
11.    public function dataSekolahAsal();
12.
13. }
14.
15. class Siswa implements Peserta {
16.
17.     public string $nama = "Kiki";
18.     public string $formasi = "SI";
19.
20.     public function dataFormasi() {
21.         return $this->formasi;
22.     }
23.
24.     public function dataDiri() {
25.         return $this->nama;
26.     }
27.
28.     public function dataOrtu() {
29.         //
30.     }
31.
32.     public function dataSekolahAsal() {
```

```
33.      //
34.    }
35.
36. }
37.
38. class Mahasiswa implements Peserta {
39.
40.     public string $nama = "Zizi";
41.     public string $formasi = "SI";
42.
43.     public function dataFormasi() {
44.         return $this->formasi;
45.     }
46.
47.     public function dataDiri() {
48.         return $this->nama;
49.     }
50.
51.     public function dataOrtu() {
52.         //
53.     }
54.
55.     public function dataSekolahAsal() {
56.         //
57.     }
58. }
59.
60. $ssw = new Siswa;
61. echo "<p>" . $ssw->dataDiri() . "</p>";
62. echo "<p>" . $ssw->dataFormasi() . "</p>";
63.
64. $mhs = new Mahasiswa;
65. echo "<p>" . $mhs->dataDiri() . "</p>";
66. echo "<p>" . $mhs->dataFormasi() . "</p>";
```

Class bisa implements banyak Interface

Sebuah *class* bisa *implements* ke banyak *interface* sekaligus dan menggunakan semua *method* yang terdapat pada *interface-interface* tersebut.

```
1. <?php
2.
3. interface Formasi {
4.     public function dataFormasi();
5. }
6.
7. interface Peserta {
8.
9.     public function dataDiri();
10.    public function dataOrtu();
11.    public function dataSekolahAsal();
12.
```

```
13.     }
14.
15.     class Siswa implements Formasi, Peserta {
16.
17.         public string $nama = "Kiki";
18.         public string $formasi = "SI";
19.
20.         public function dataFormasi() {
21.             return $this->formasi;
22.         }
23.
24.         public function dataDiri() {
25.             return $this->nama;
26.         }
27.
28.         public function dataOrtu() {
29.             //
30.         }
31.
32.         public function dataSekolahAsal() {
33.             //
34.         }
35.
36.     }
37.
38.     class Mahasiswa implements Formasi, Peserta {
39.
40.         public string $nama = "Zizi";
41.         public string $formasi = "SI";
42.
43.         public function dataFormasi() {
44.             return $this->formasi;
45.         }
46.
47.         public function dataDiri() {
48.             return $this->nama;
49.         }
50.
51.         public function dataOrtu() {
52.             //
53.         }
54.
55.         public function dataSekolahAsal() {
56.             //
57.         }
58.     }
59.
60.     $ssw = new Siswa;
61.     echo "<p>" . $ssw->dataDiri() . "</p>";
62.     echo "<p>" . $ssw->dataFormasi() . "</p>";
63.
64.     $mhs = new Mahasiswa;
65.     echo "<p>" . $mhs->dataDiri() . "</p>";
66.     echo "<p>" . $mhs->dataFormasi() . "</p>";
```

5.3 Latihan

Rancanglah sebuah program singkat dari sebuah kasus sederhana dalam kehidupan sehari-hari menggunakan metode *polymorphism* baik dalam bentuk *Abstract class* maupun *Interface*.

MODUL 6

NAMESPACE

6.1 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

- Memahami penggunaan *namespace* dalam konsep OOP pada bahasa pemrograman PHP.
- Memahami dan Merancang pola desain program menggunakan perintah *namespace*.

6.2 Praktikum

Sejarah penggunaan *namespace* bermula saat orang-orang ingin membagikan kodenya agar dapat dipakai oleh orang banyak. Namun yang terjadi selanjutnya adalah munculnya konflik karena penamaan *class* yang sama, sedang di PHP, nama *class* harus unik. Jadi ketika kita memakai *third party library* yang memiliki *class User* , kita tidak bisa membuat *class* dengan nama *User* juga. Awalnya orang-orang mengakalinya dengan menambahkan *underscore* sehingga penamaan *class* menjadi sangat panjang. Namun untungnya sejak versi 5.3, PHP telah mendukung penggunaan *namespace* (Medium).

Namespace merupakan perintah khusus yang digunakan untuk menghindari konflik dari penamaan *class* yang sama. Dengan menggunakan perintah *namespace*, pembuat program lebih leluasa menyusun struktur program walaupun nanti terdapat nama *class* yang sama ketika dipanggil secara bersamaan. Untuk lebih memahami perintah *namespace*, perhatikan contoh berikut ini :

```
app
|—— Mahasiswa.php
inc
|—— Mahasiswa.php
layout
|—— Mahasiswa.php
index.php
```


Pada struktur diatas, saya punya tiga *folder* yaitu **app**, **inc** dan **layout**. Masing-masing *folder* mempunyai *file* dengan nama yaitu **Mahasiswa.php** dengan *class* Mahasiswa didalamnya, ketiga *file* ini akan saya panggil kedalam file **index.php**

app/Mahasiswa.php

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     public function __construct() {
6.         echo "Saya app/Mahasiswa.php";
7.     }
8. }
```

inc/Mahasiswa.php

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     public function __construct() {
6.         echo "Saya inc/Mahasiswa.php";
7.     }
8. }
```

layout/Mahasiswa.php

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     public function __construct() {
6.         echo "Saya layout/Mahasiswa.php";
7.     }
8. }
```

index.php

```
1. <?php
2.
3. include "app/Mahasiswa.php";
4. include "inc/Mahasiswa.php";
5. include "layout/Mahasiswa.php";
6.
7. $app = new Mahasiswa;
8. $inc = new Mahasiswa;
9. $layout = new Mahasiswa;
```

Ketika file **index.php** dijalankan, dimana *file* tersebut memanggil *file* Mahasiswa.php dari masing-masing *folder*, maka dapat dipastikan layar menampilkan pemberitahuan **error**. Hal ini karena *class* mahasiswa tidak bisa ditampilkan bersamaan.

Nah, sekarang bagaimana mengatasi hal ini? Disinilah peran *Namespace*.

app/Mahasiswa.php

```
1. <?php
2. namespace App;
3.
4. class Mahasiswa {
5.
6.     public function __construct() {
7.         echo "Saya app/Mahasiswa.php";
8.     }
9. }
```

inc/Mahasiswa.php

```
1. <?php
2. namespace Inc;
3.
4. class Mahasiswa {
5.
6.     public function __construct() {
7.         echo "Saya inc/Mahasiswa.php";
8.     }
9. }
```

layout/Mahasiswa.php

```
1. <?php
2. namespace Layout;
3.
4. class Mahasiswa {
5.
6.     public function __construct() {
7.         echo "Saya layout/Mahasiswa.php";
8.     }
9. }
```

index.php

```
10. <?php
11.
12. include "app/Mahasiswa.php";
13. include "inc/Mahasiswa.php";
14. include "layout/Mahasiswa.php";
15.
16. $app = new App\Mahasiswa;
17. $inc = new Inc\Mahasiswa;
18. $layout = new Layout\Mahasiswa;
```

Setelah ditambahkan perintah *namespace*, ketika dipanggil dalam *file* yang sama yaitu **index.php**, walaupun dengan nama *class* yang sama, maka sistem tetap berjalan dengan baik.

6.3 Latihan

Perbaikilah struktur program yang telah saudara rancang pada praktikum sebelumnya, gunakan perintah *namespace* pada file yang diperlukan.

MODUL 7

COMPOSER

7.1 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

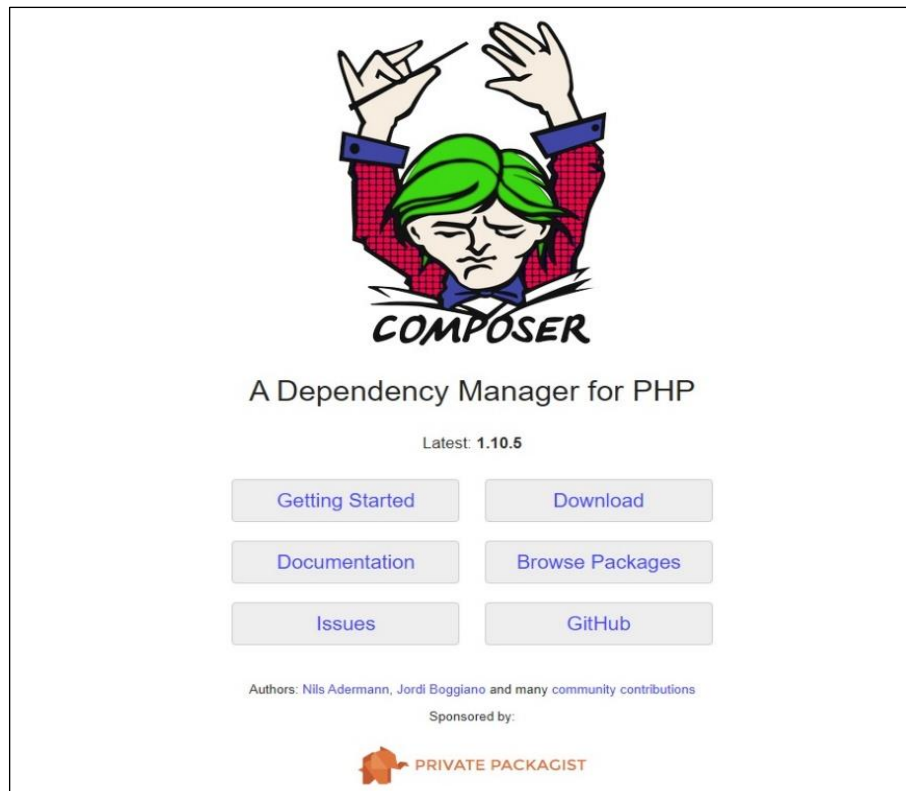
- Memahami penggunaan *composer* pada bahasa pemrograman PHP.
- Memahami penggunaan *Git* pada bahasa pemrograman PHP.
- Memahami dan Merancang program menggunakan perintah-perintah yang disediakan oleh *Composer*.
- Memahami dan Merancang program menggunakan perintah-perintah yang disediakan oleh *Git*.

7.2 Praktikum

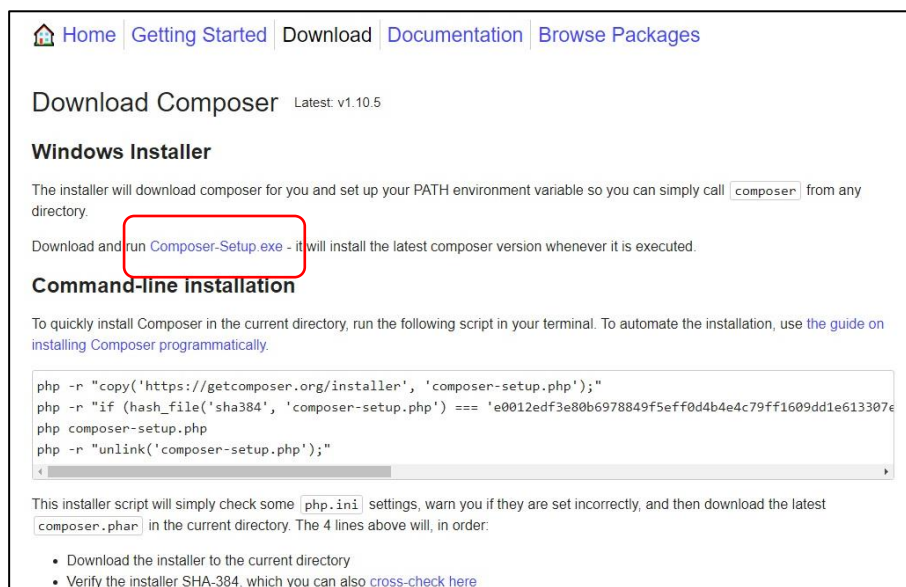
Composer merupakan sebuah *project open source* yang dimotori oleh Nils Adermann dan Jordi Boggiano. *Project composer* ini di-hosting di github (<https://github.com/composer/composer>) tercatat sejak tanggal 3 April 2011 dan masih aktif sampai sekarang. *Composer* adalah alat manajemen *dependency* pada PHP, *Composer* memungkinkan untuk membuat *library* pada *project* dan *composer* sendiri akan meng-*install* atau meng-*update* secara otomatis.

Git adalah salah satu sistem pengontrol versi (*Version Control System*) pada proyek perangkat lunak yang diciptakan oleh Linus Torvalds. Pengontrol versi bertugas mencatat setiap perubahan pada *file* proyek yang dikerjakan oleh banyak orang maupun sendiri. *Git* dikenal juga dengan *distributed revision control* (VCS terdistribusi), artinya penyimpanan database *Git* tidak hanya berada dalam satu tempat saja. Jadi dengan *Git* ini kita dengan mudah untuk mengontrol seluruh perubahan-perubahan yang kita lakukan terhadap proyek yang sedang kita kembangkan. Berikut ini kita akan bahas secara ringan, mulai dari Instalasi sampai penggunaan *Composer* dan *Git*.

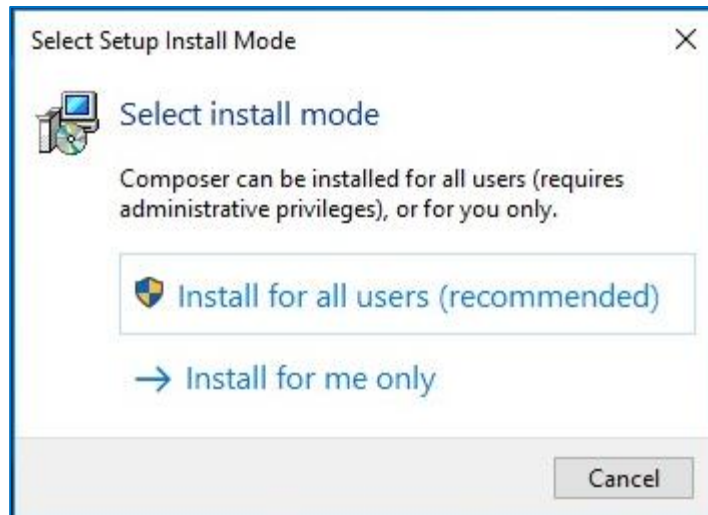
- Untuk mendapatkan aplikasi *composer*, silakan langsung ke web resminya <https://getcomposer.org/>



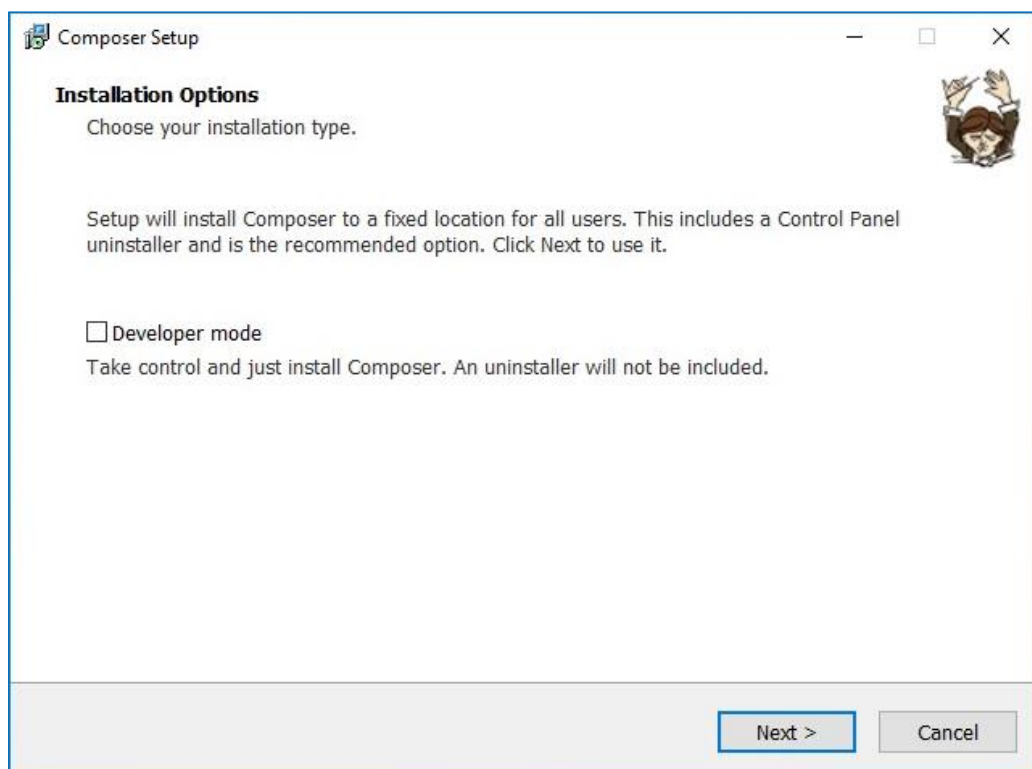
Klik *download* pada menu yang disediakan untuk diarahkan ke halaman *download*, ukuran *file composer* cukup kecil dan tentunya *update* sesuai dengan perkembangan terkini.



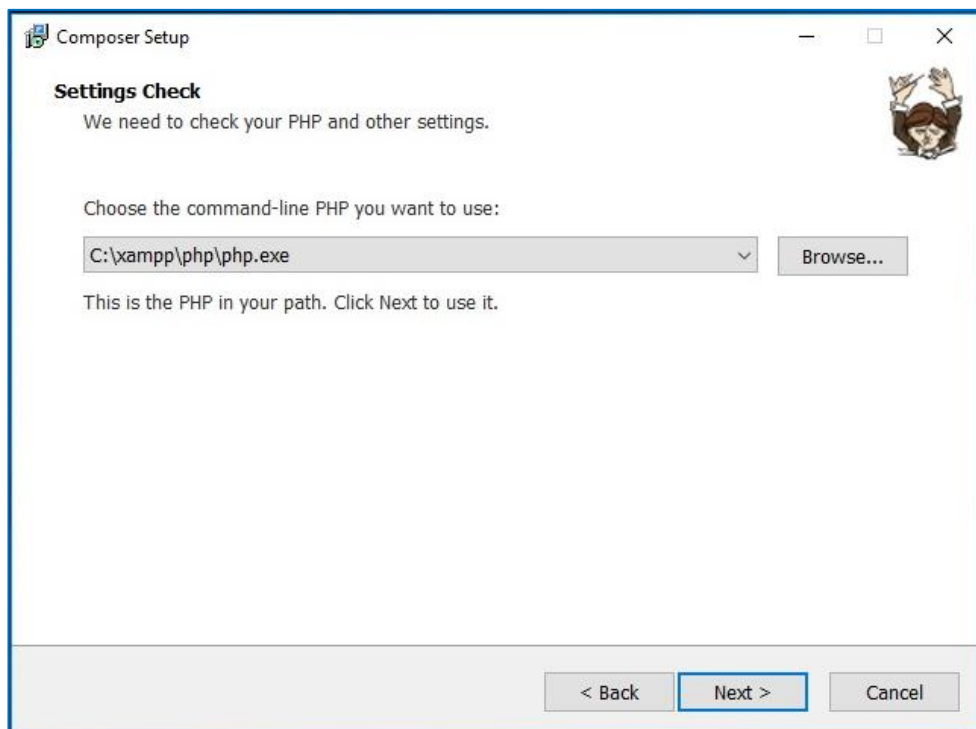
Klik pada link **Composer-Setup.exe**, sebagaimana gambar diatas diatas pada bagian yang dilingkari.



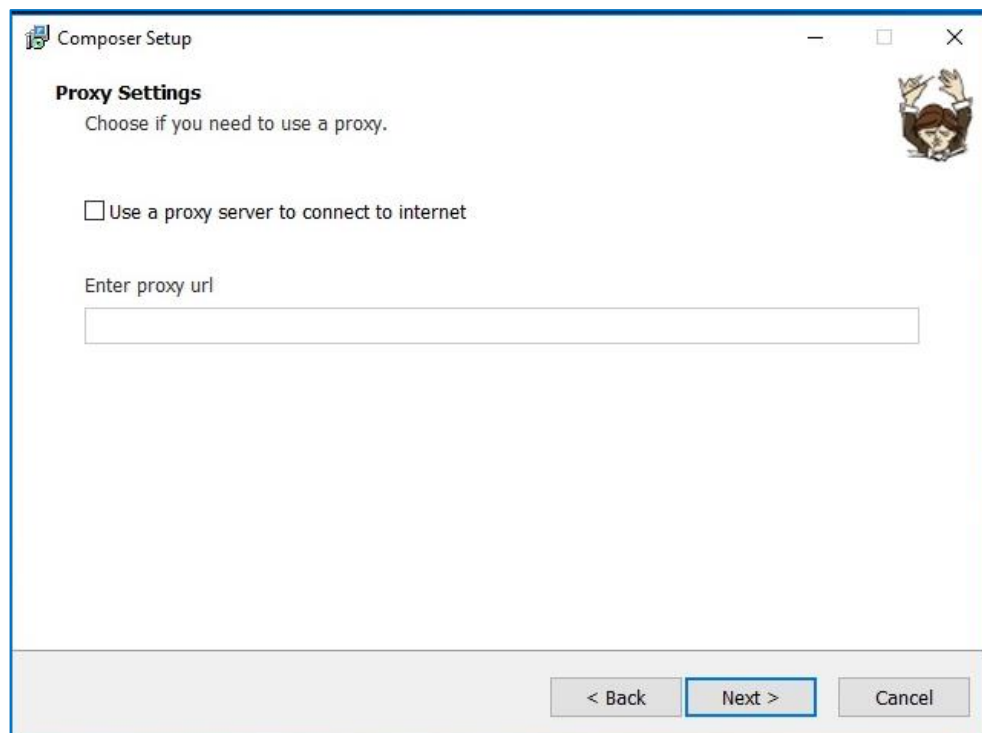
Selanjutnya lakukan instalasi dengan *double* Klik pada *file* yang telah di *download* sebelumnya, selanjutnya pilih menu *Install for all users (recommended)* pada saat halaman pemilihan mode *install*.



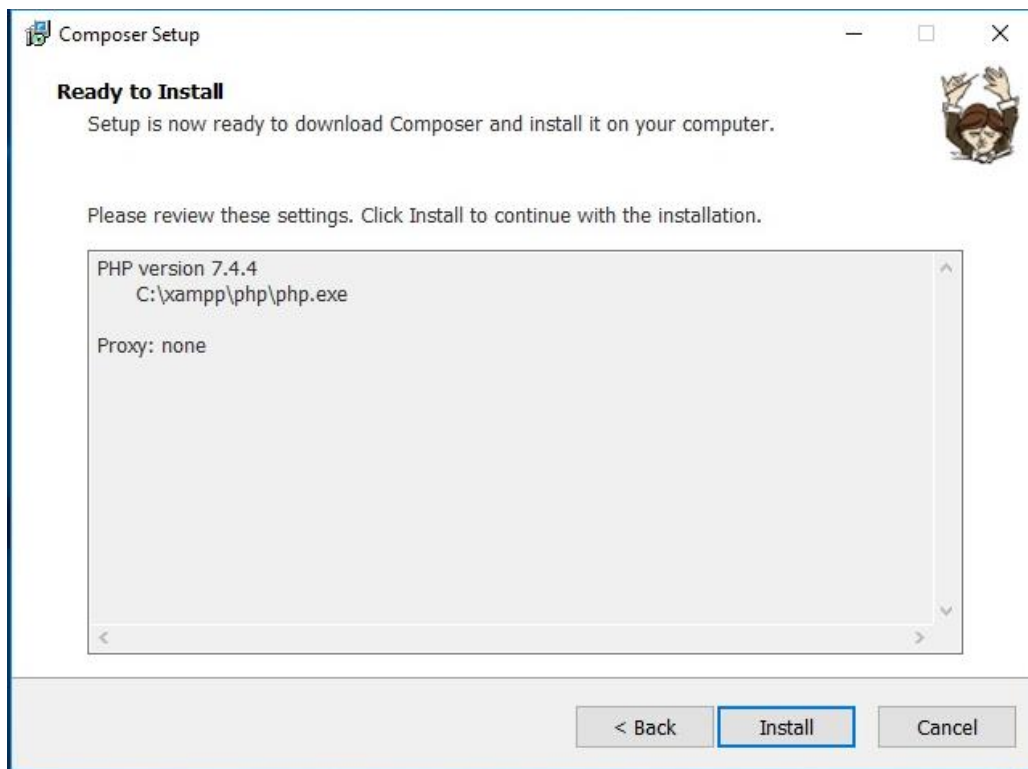
Halaman diatas menampilkan tipe instalasi, Klik **next** menuju halaman instalasi selanjutnya.



Halaman diatas bagian penting dari Instalasi *Composer*. *Composer* memeriksa ketersediaan PHP yang terdapat pada komputer kita. *Path* diatas merupakan standar bawaan yang terdapat pada XAMPP. Nah, karena pada Praktikum kita ini menggunakan XAMPP, *Composer* sudah otomatis mendeteksi PHP yang terpasang.



Untuk halaman *proxy*, silakan di-klik **Next** saja untuk *proxy* otomatis.



Terakhir anda siap untuk proses pemasangan *composer*, Klik **Install** untuk memasang *composer* di komputer anda. Waktu yang dibutuhkan cukup singkat, sesuai dengan ukuran *file* instalasi yang diberikan.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Faiz>composer

Composer
Composer version 1.10.5 2020-04-10 11:44:22

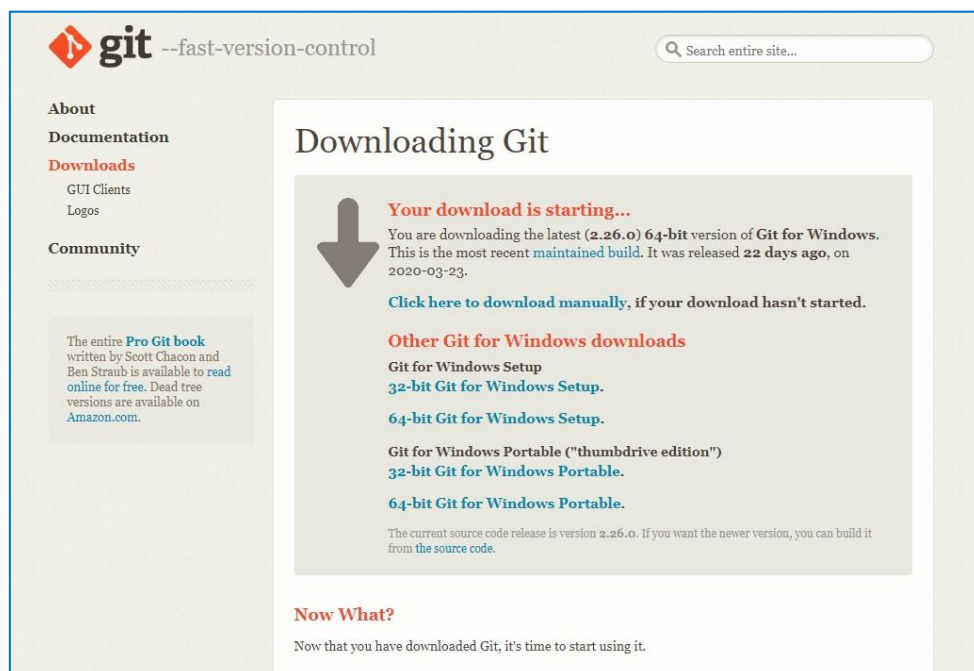
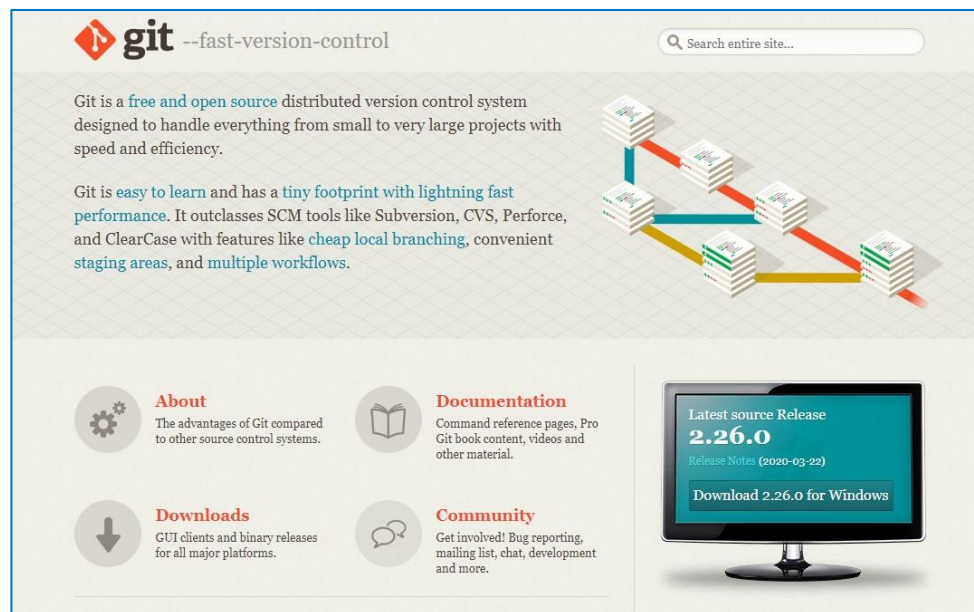
Usage:
  command [options] [arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
  --ansi                   Force ANSI output
  --no-ansi                Disable ANSI output
  -n, --no-interaction      Do not ask any interactive question
  --profile                Display timing and memory usage information
  --no-plugins              Whether to disable plugins.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  --no-cache               Prevent use of the cache
  -v|vv|vvv, --verbose     Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and
                             3 for debug

Available commands:
```

Jika sudah selesai, buka *Command Prompt* di Windows anda, kemudian ketik-kan perintah `composer` . Jika tidak ada masalah, maka sistem akan menampilkan informasi *composer* yang telah terpasang.

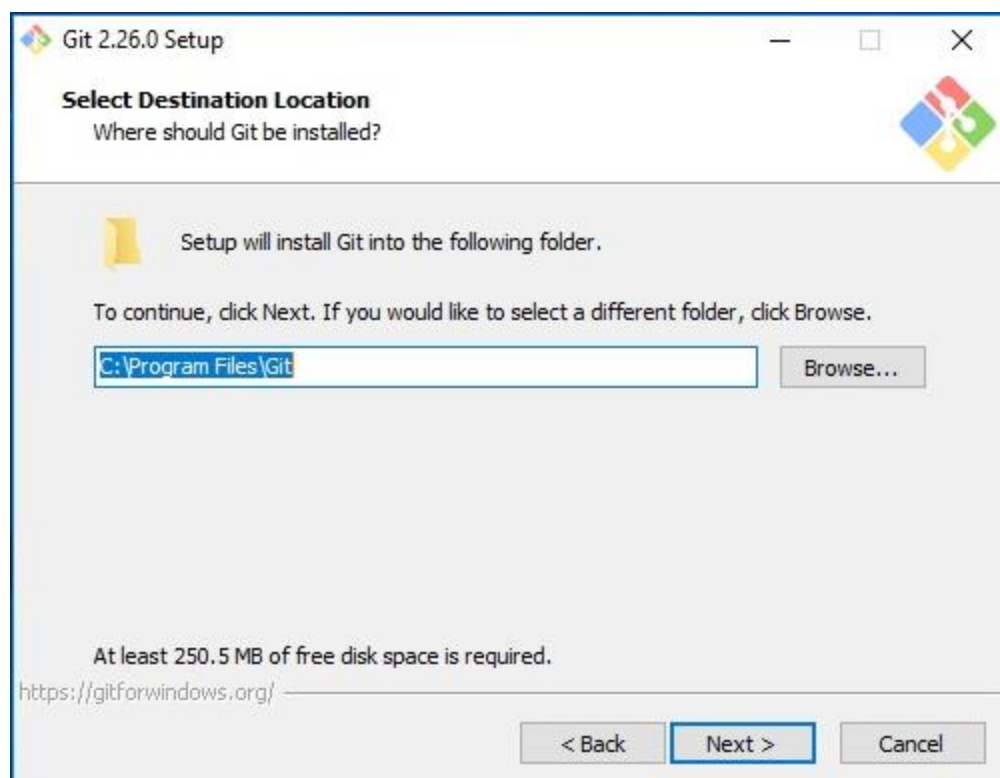
Setelah *composer* terpasang, selanjutnya kita akan memasang *Git*. File instalasi bisa ada dapatkan langsung dari web resminya. <https://git-scm.com/>

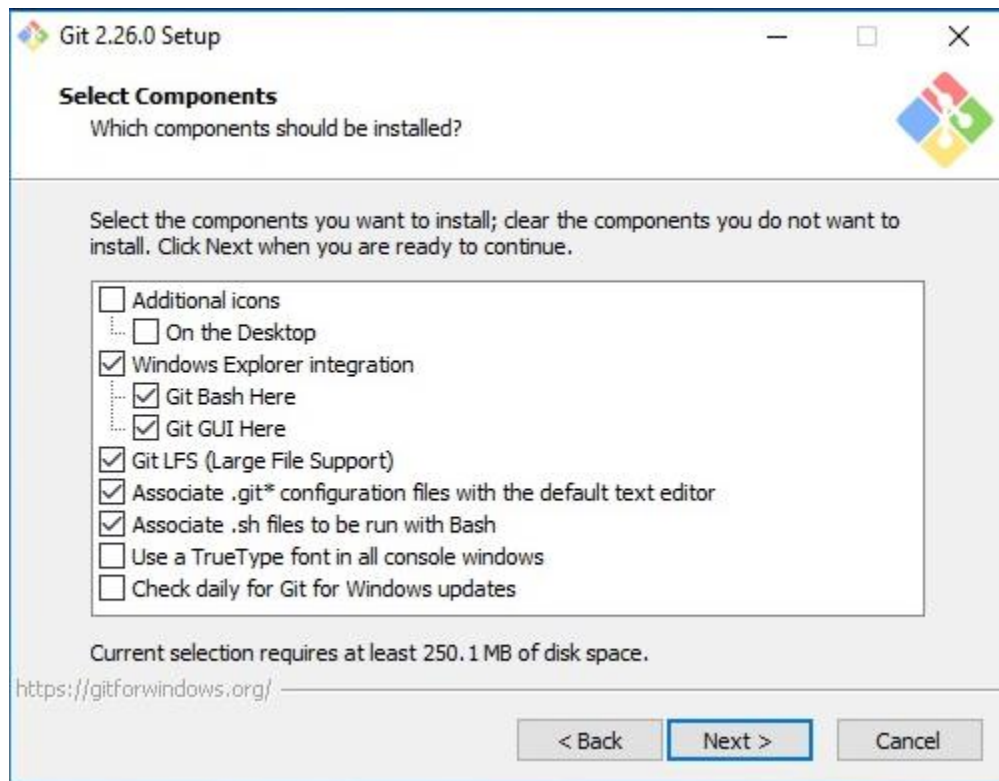


Klik pada menu **Download** yang disediakan, anda akan diarahkan ke halaman *download* dan *download* akan dimulai secara otomatis. Silakan tunggu hingga proses *download* selesai. Setelah selesai, *double* klik pada *file* instalasi dan proses instalasi akan dimulai.

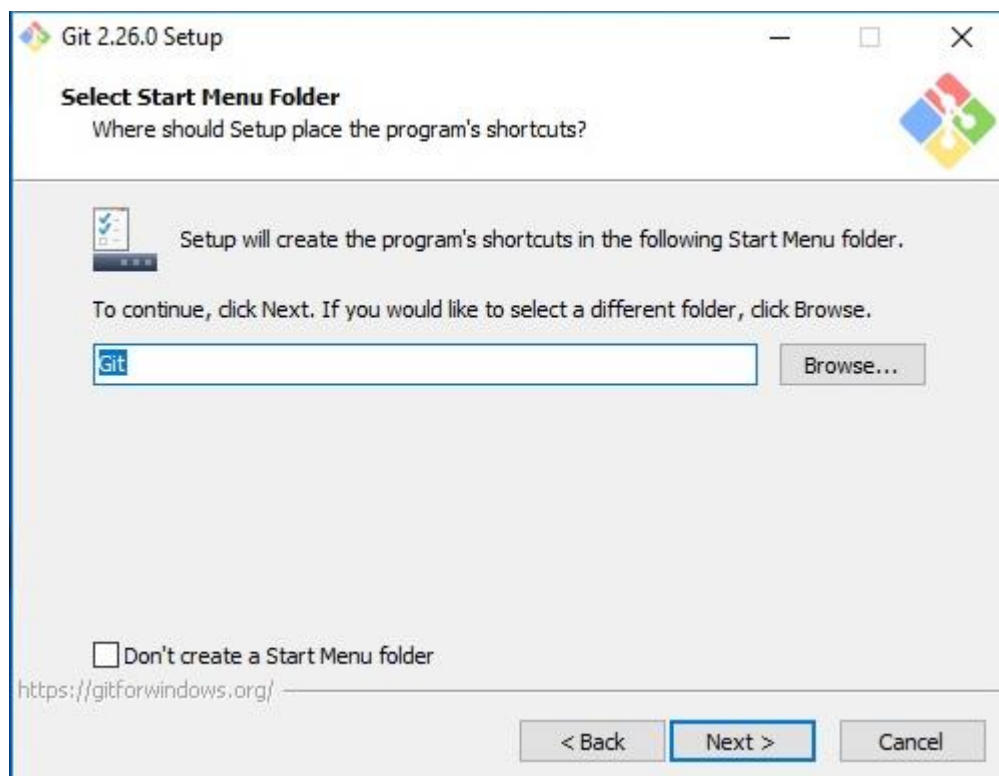


Halaman pertama instalasi menampilkan informasi dari *Git* yang akan kita pasang, klik **Next** untuk melanjutkan instalasi.

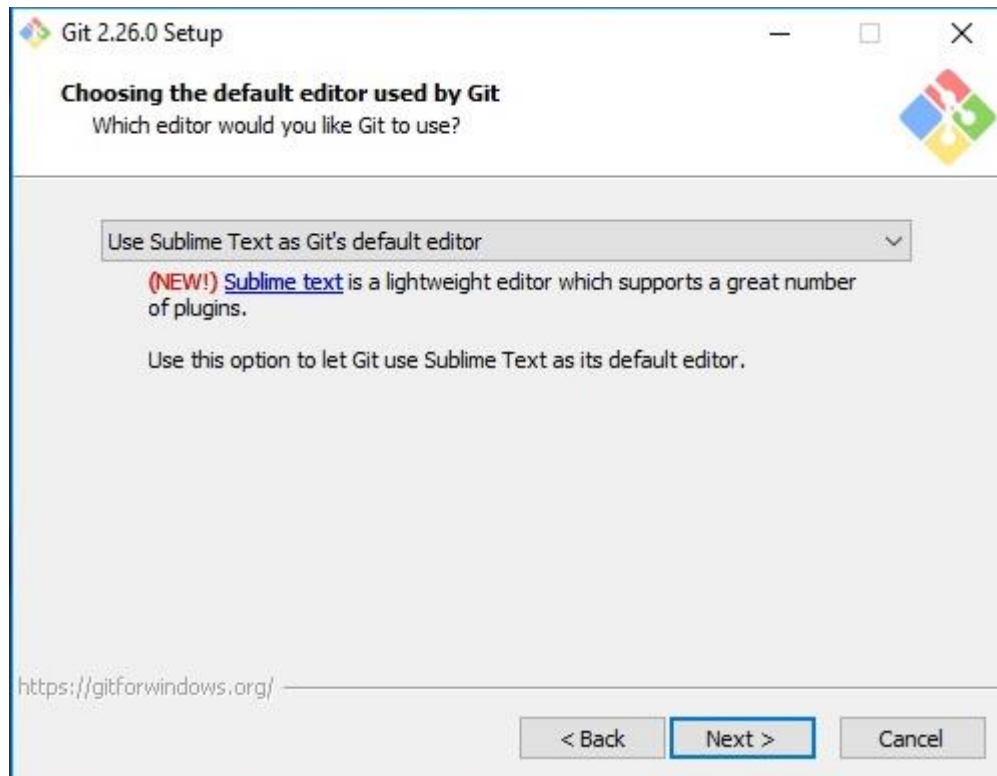




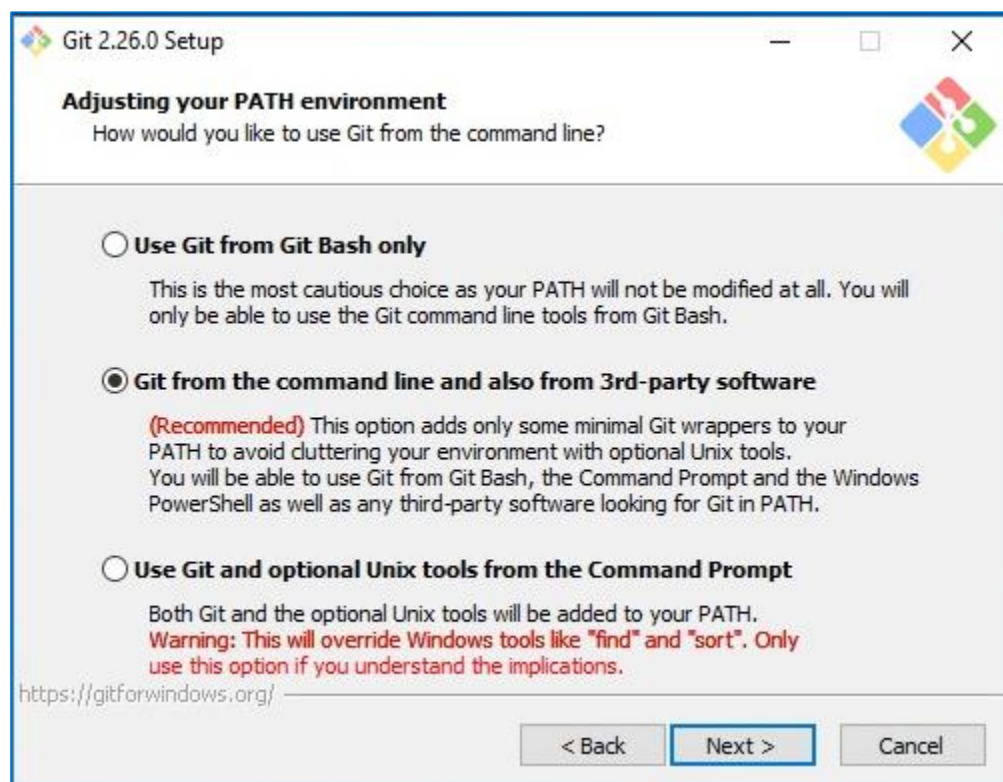
Anda bisa saja menggunakan pilihan bawaan saat pemilihan komponen seperti ditampilkan pada gambar diatas. Klik **Next** untuk melanjutkan instalasi.

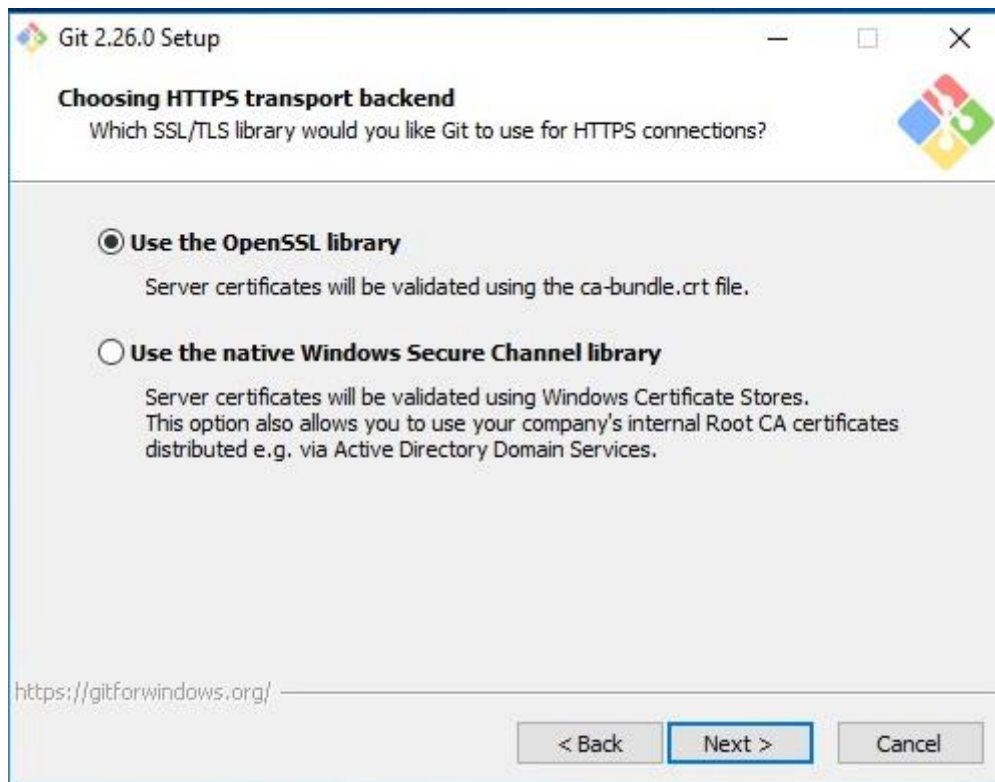


Gunakan opsi bawaan saja untuk *folder* di *Start Menu*, Kemudian klik **Next**.

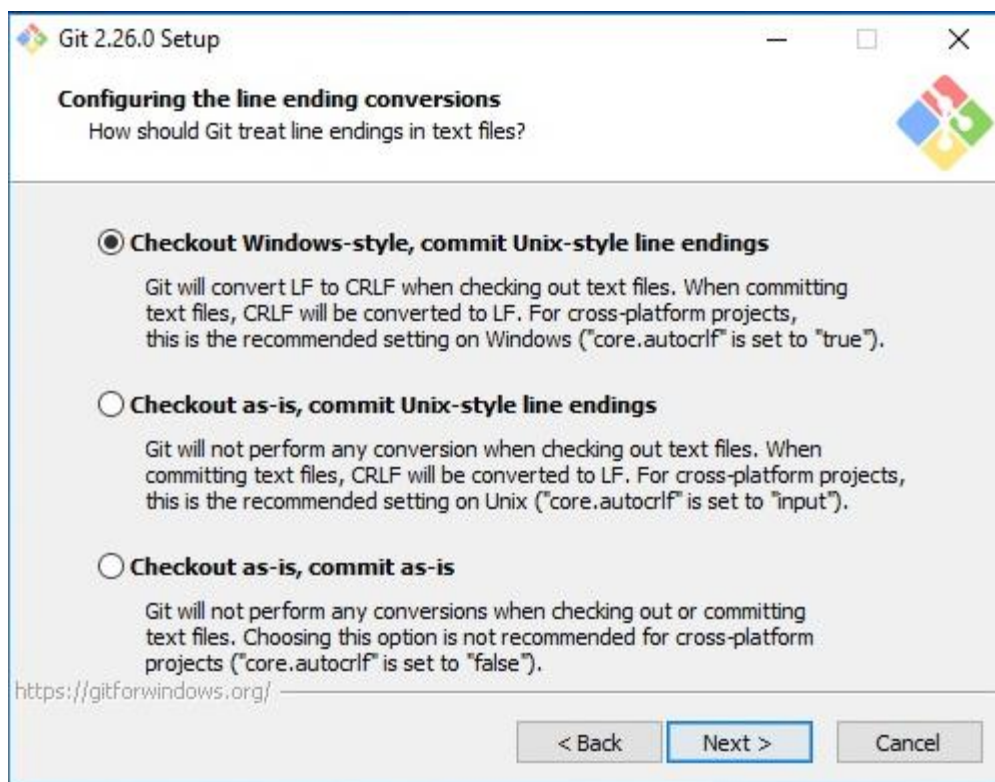


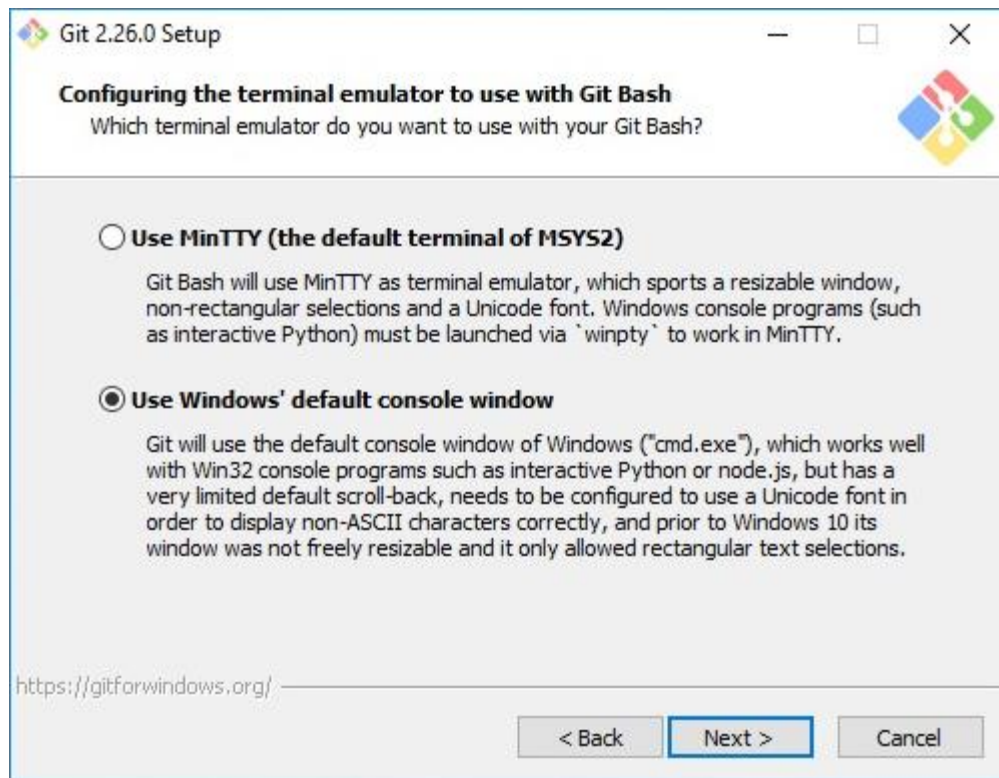
Selanjutnya Text Editor bawaan yang akan digunakan oleh *Git*, karena pada praktikum ini kita menggunakan *Sublime Text 3*, pilih text editor tersebut pada opsi yang diberikan. Klik **Next** untuk halaman berikutnya.





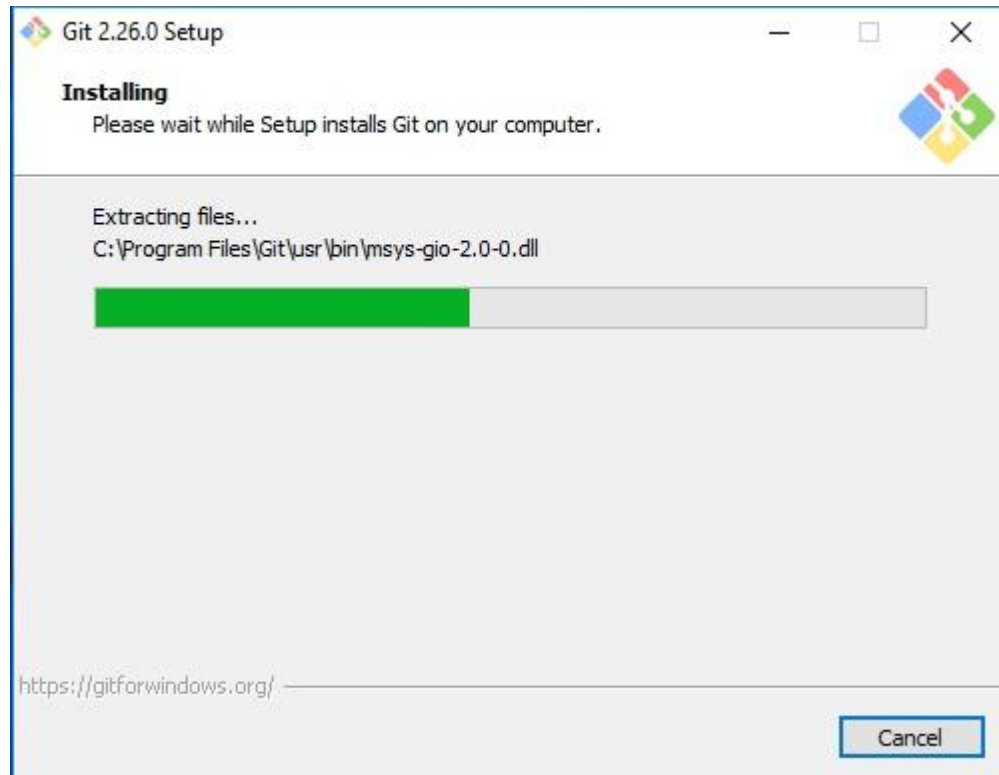
Untuk koneksi https, gunakan OpenSSL saja sebagai opsi bawaan, klik **Next** untuk halaman berikutnya.



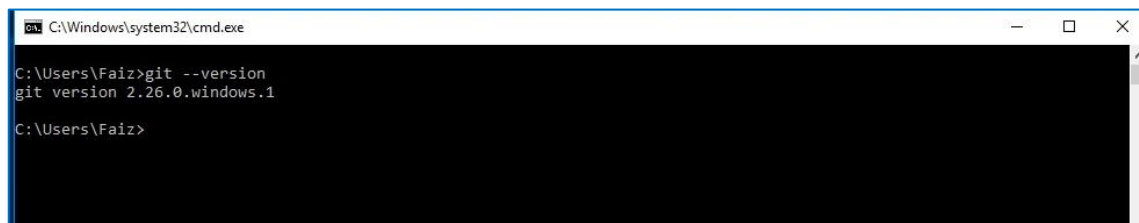


Gunakan pilihan *default console* saja yaitu *Command Prompt* yang telah tersedia di Windows masing-masing, klik **Next** menuju halaman berikutnya.





Sudah siap untuk pemasang, silakan tunggu hingga proses instalasi selesai. Jika telah selesai, kembali buka *Command Prompt* dari Windows anda dan ketikkan perintah `git --version`. Perintah tersebut akan menampilkan versi git yang sudah terpasang.

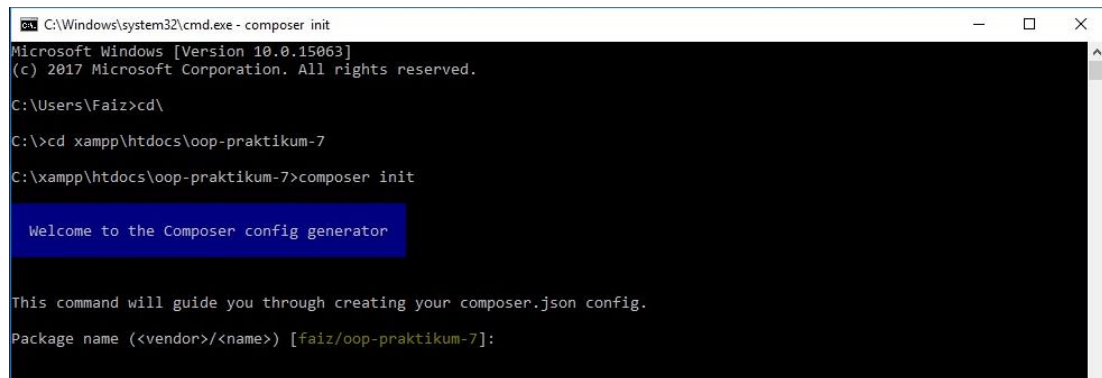


Memulai Project

Setelah *composer* terpasang, saatnya kita menggunakannya pada *project* yang akan kita bangun.

- Masuk ke *folder* sistem yang akan kita rancang dan ketikkan perintah

```
composer init
```



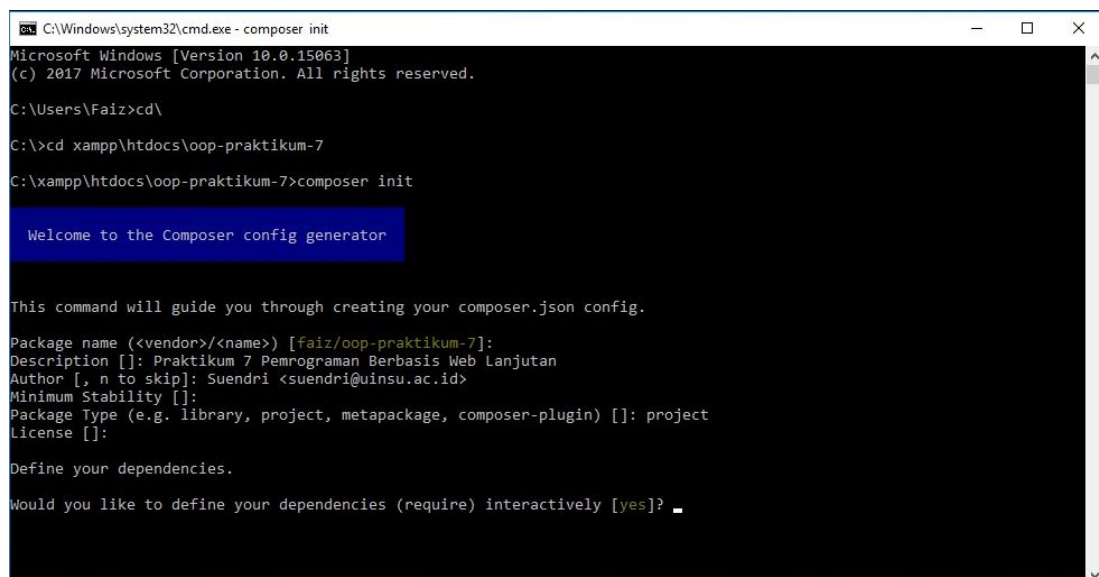
```
C:\Windows\system32\cmd.exe - composer init
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Faiz>cd\
C:\>cd xampp\htdocs\oop-praktikum-7
C:\xampp\htdocs\oop-praktikum-7>composer init

Welcome to the Composer config generator

This command will guide you through creating your composer.json config.
Package name (<vendor>/<name>) [faiz/oop-praktikum-7]:
```

- Isikan *package name*, atau enter saja jika anda setuju dengan *default*. Kemudian isian selanjutnya seperti pada gambar dibawah ini :



```
C:\Windows\system32\cmd.exe - composer init
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

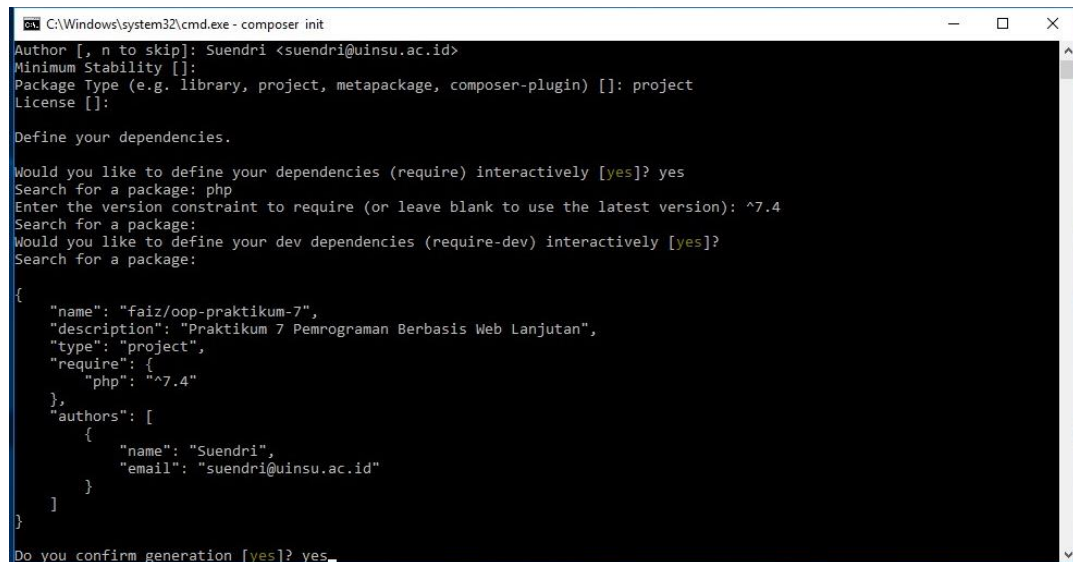
C:\Users\Faiz>cd\
C:\>cd xampp\htdocs\oop-praktikum-7
C:\xampp\htdocs\oop-praktikum-7>composer init

Welcome to the Composer config generator

This command will guide you through creating your composer.json config.
Package name (<vendor>/<name>) [faiz/oop-praktikum-7]:
Description []: Praktikum 7 Pemrograman Berbasis Web Lanjutan
Author [, n to skip]: Suendri <suendri@uinsu.ac.id>
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []: project
License []:

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]? _
```

```
C:\Windows\system32\cmd.exe - composer init
Author [, n to skip]: Suendri <suendri@uinsu.ac.id>
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []: project
License []:

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]? yes
Search for a package: php
Enter the version constraint to require (or leave blank to use the latest version): ^7.4
Search for a package:
Would you like to define your dev dependencies (require-dev) interactively [yes]?
Search for a package:

{
  "name": "faiz/ooop-praktikum-7",
  "description": "Praktikum 7 Pemrograman Berbasis Web Lanjutan",
  "type": "project",
  "require": {
    "php": "^7.4"
  },
  "authors": [
    {
      "name": "Suendri",
      "email": "suendri@uinsu.ac.id"
    }
  ]
}

Do you confirm generation [yes]? yes
```

- c. Proses ini juga menawarkan *library* yang akan di-*install*, silakan sesuai dengan sistem yang dirancang. Nah, kita dapat dari mana *library-library* ini? Yah jawaban salah satunya adalah Github di <https://github.com/>. Github akan kita bahas secara terpisah di Modul berikutnya. Jika proses diatas sudah selesai, kita akan mendapat sebuah *folder* baru, namanya **vendor**. Didalam *folder* inilah semua *library* yang kita butuhkan akan tersusun rapi.

Name	Date modified	Type	Size
app	4/14/2020 12:52 AM	File folder	
vendor	4/14/2020 12:49 AM	File folder	
composer.json	4/14/2020 12:49 AM	JSON File	1 KB
composer.lock	4/14/2020 12:49 AM	LOCK File	1 KB
index.php	4/14/2020 12:52 AM	PHP File	1 KB

Oke, demikian untuk modul praktikum 7 ini, silakan pelajari video tutorial sebagai pelengkap dari modul ini pada link yang sudah dibagikan.

7.3 Latihan

Dosen pengampu memberikan Tugas 1 dan Quiz 1, Silakan kerjakan dengan sungguh-sungguh.

MODUL 8

REPOSITORY

8.1 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

- a. Memahami definisi *repository* sebagai media penyimpanan kode program berbasis *cloud* menggunakan Github.
- b. Memahami penggunaan aplikasi Github berbasis desktop.
- c. Memahami langkah upload *source code* program ke *repository* Github.
- d. Memahami dan merancang program dengan cara kolaborasi dengan pengguna *repository* lainnya.

8.2 Praktikum

Repository adalah direktori penyimpanan yang berfungsi untuk mengumpulkan, mengatur dan menyebarkan data dalam bentuk digital, baik secara online maupun secara offline. Pada bidang manajemen proyek Sistem Informasi, anda bisa menyimpan apa pun yang berkaitan dengan proyek yang sedang dibuat, misalnya file kode, gambar, video atau audio.

Github merupakan sebuah aplikasi berbasis web yang memberikan layanan repository berupa penyimpanan kode program berbasis *cloud*, baik penyimpanan untuk tujuan komersial berbayar maupun penyimpanan gratis. Github juga merupakan salah satu *Version Control System* (VCS) yang paling populer selain Gitlab, BitBucket, SourceForge dan sebagainya. *Version Control* adalah sebuah sistem yang merekam perubahan-perubahan dari sebuah berkas atau sekumpulan berkas dari waktu ke waktu sehingga anda dapat menilik kembali versi khusus yang dilakukan perubahan suatu saat nanti pada saat dibutuhkan. Github bisa diakses secara personal oleh masing-masing pengembang perangkat lunak, selain itu Github bisa juga untuk kerjasama dalam tim, tanpa terbatas waktu dan tempat. Repository Github mendukung semua bahasa pemrograman, anda bebas menggunakan bahasa pemrograman dan aplikasi yang anda gunakan.



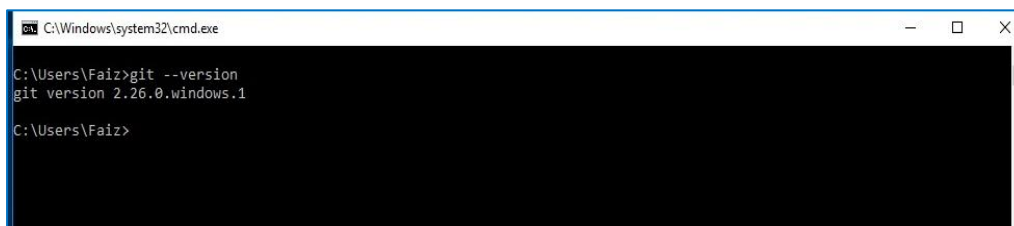
Langkah penggunaan Github

Untuk menggunakan Github bisa dilakukan menggunakan 2 cara :

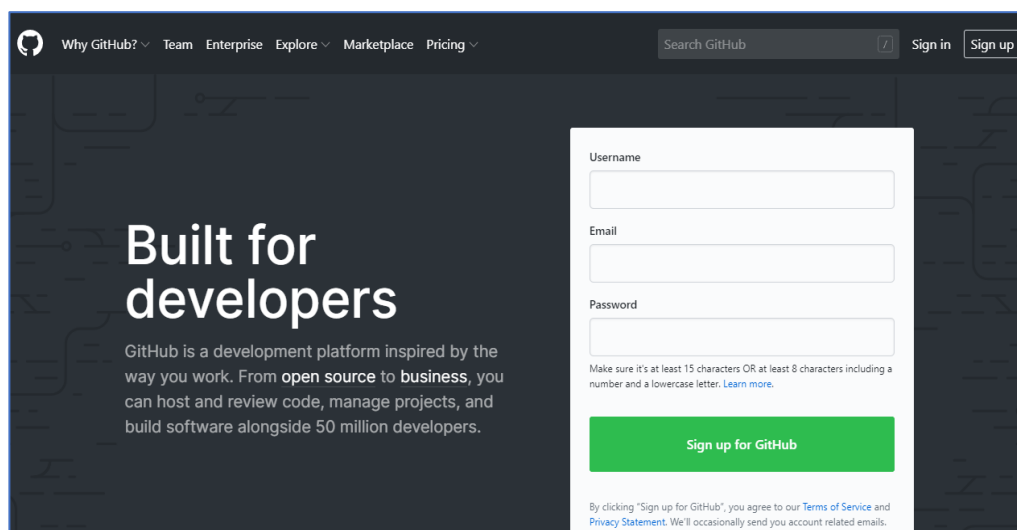
1. Menggunakan *Command Line*

Untuk menggunakan *Command Line*, pastikan anda sudah memasang aplikasi Git yang sudah kita bahas pada Modul 7 sebelumnya. Periksa aplikasi Git yang sudah terpasang menggunakan perintah berikut dari command prompt.

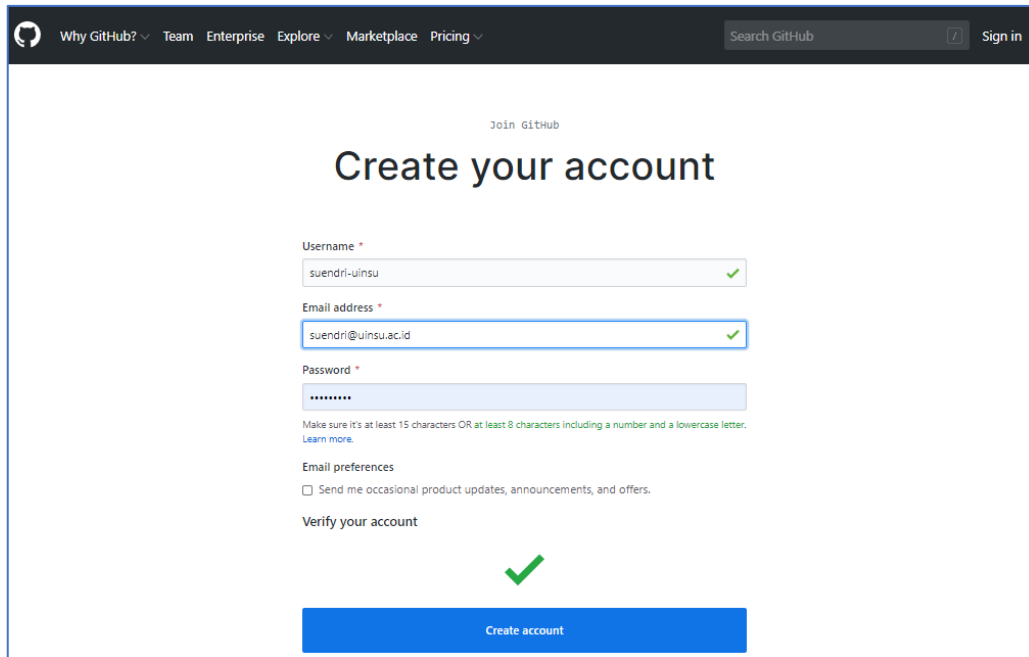
```
git --version
```



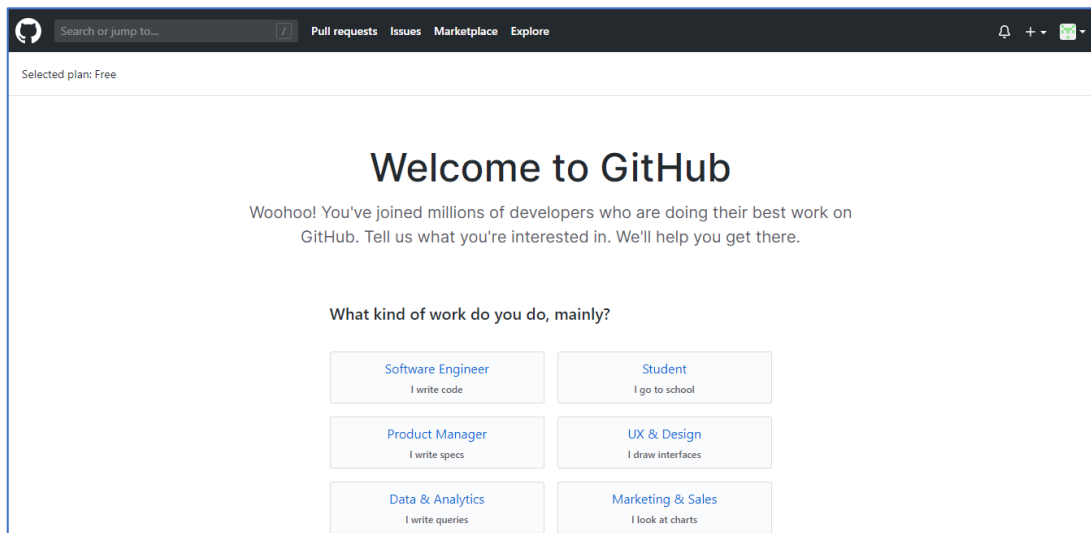
Selanjutnya, buat akun di Github. Pastikan komputer anda terkoneksi dengan internet dan buka alamat resmi Github di <http://github.com>.



Pilih menu **Signup** dan isi data input sesuai yang diminta, data yang anda masukkan akan diperiksa secara langsung oleh sistem Github.



Setelah data akun diterima oleh Github, ada akan diberikan pilihan jenis pekerjaan yang sedang anda lakukan, boleh pilih salah satunya atau pilih *Student* agar sesuai dengan kondisi saat ini.






Untuk pilihan pengalaman programing, pilihan opsi **A little** saja. Namun anda tentunya diperbolehkan untuk memilih opsi yang lain.

How much programming experience do you have?

None I don't program at all	A little I'm new to programming
A moderate amount I'm somewhat experienced	A lot I'm very experienced

Kemudian opsi berikutnya, pilih **Host a Project**.

What do you plan to use GitHub for?
(Select up to 3)

 Learn to code	 Learn Git and GitHub	 Host a project (repository)
----------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------

Terakhir pada opsi ketertarikan, isikan beberapa kata kunci dan klik **Complete Setup**.

I am interested in:

php × mysql × framework × laravel × bootstrap ×

languages, frameworks, industries

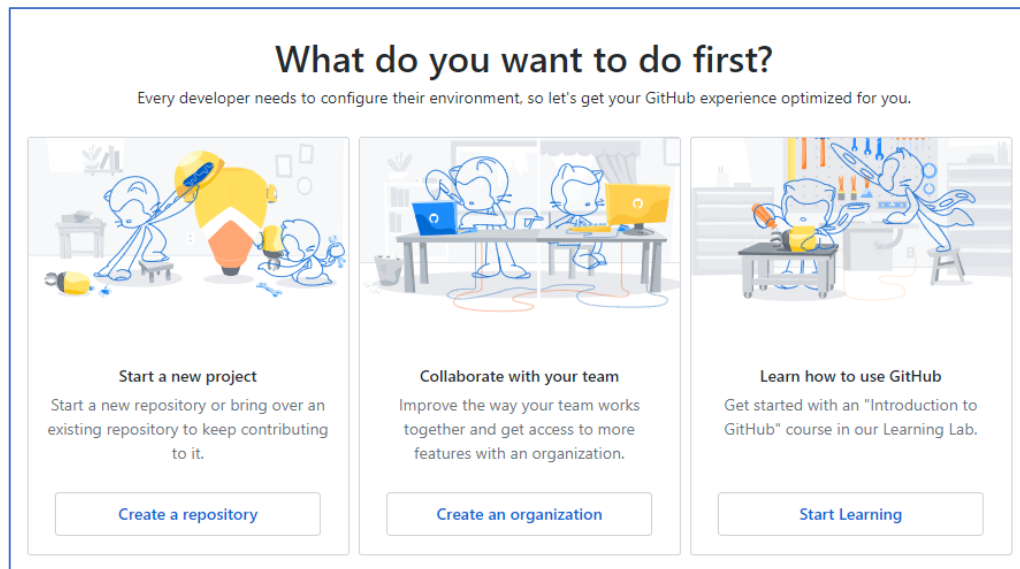
We'll connect you with communities and projects that fit your interests.

For example: duckduckgo uwp blockchain

Complete setup

Terakhir anda diminta untuk verifikasi email, agar pengguna email tersebut benar-benar sah dan mempunyai kendali terhadap email yang didaftarkan. Untuk login berikutnya anda bisa menggunakan email yang telah diverifikasi atau jika suatu saat anda lupa password, reset password akan dikirimkan pada email yang telah diverifikasi tersebut.

Setelah verifikasi akun berhasil, anda akan diarahkan ke akun Github yang baru saja anda buat dan ditawarkan 3 opsi, pilih opsi **Create a Repository**



Kemudian isi form yang diberikan sesuai dengan *project* yang sedang dikerjakan. Untuk praktikum ini saya buat nama repository-nya

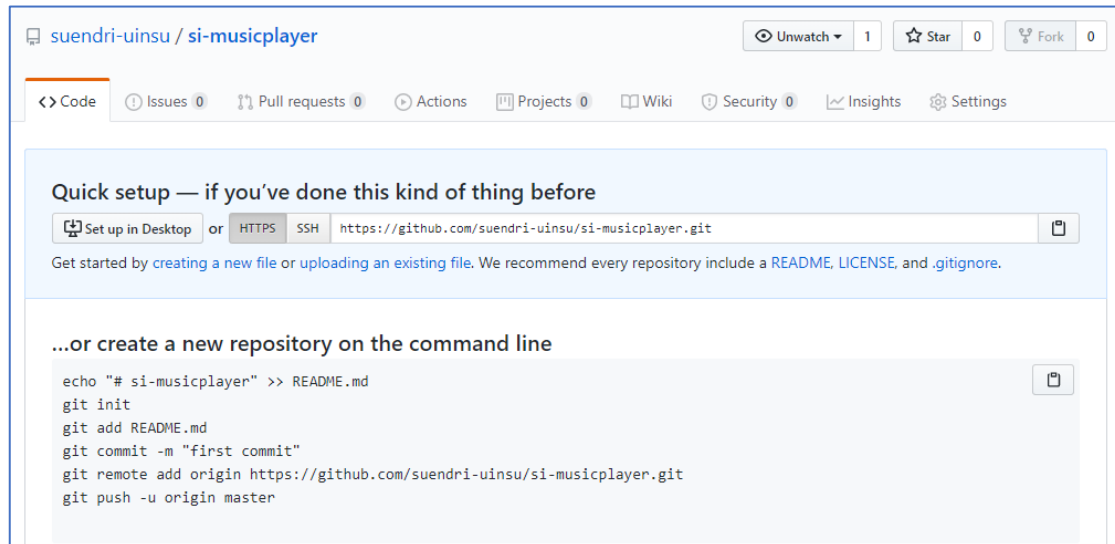
si-musicplayer

kemudian isi deskripsi *project* sesuai yang anda inginkan.

The screenshot shows the "Create a new repository" form. It includes the following fields and options:

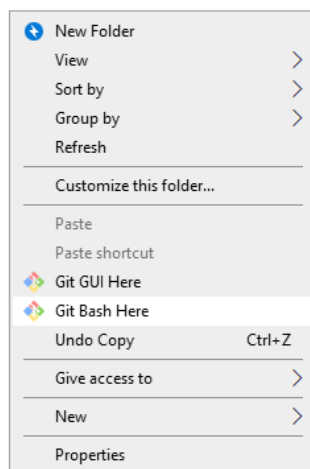
- Owner**: A dropdown menu showing "suendri-uinsu".
- Repository name ***: A text input field containing "si-musicplayer" with a green checkmark icon to its right.
- Description (optional)**: A text input field containing "Praktikum Sistem Informasi Music Player".
- Visibility**: Two radio buttons. The "Public" option is selected, with the text "Anyone can see this repository. You choose who can commit." below it. The "Private" option is unselected, with the text "You choose who can see and commit to this repository." below it.

Anda juga bisa memilih, apakah repository ini bisa dilihat oleh orang banyak (*Public*) atau hanya kita sendiri sebagai pemilik akun (*Private*). Setelah formulir isian ini anda lengkapi, klik tombol **Create repository** untuk menyelesaikan pembuatan repository ini.



Oke! Repositori **si-musicplayer** sudah berhasil kita buat, namun masih kosong karena belum dilakukan proses *upload file project* kedalam repository tersebut. Berikutnya silakan ke *folder* project, untuk praktikum 8 saya kasih nama foldernya **oop-praktikum-8** di **c:xampp/htdocs/**.

Buka folder **oop-praktikum-8** dan klik kanan dalam *folder* tersebut dan pilih menu **Git Bash Here**

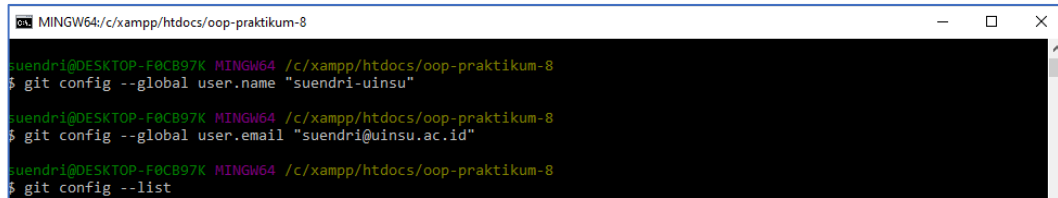


Anda akan diarahkan ke aplikasi *command prompt* sesuai dengan aplikasi *default* yang digunakan saat instalasi pada modul 7. Jika anda menggunakan aplikasi *default* lainnya, menu ini akan menampilkan sesuai dengan aplikasi yang anda pilih saat instalasi.

Hubungkan aplikasi Git dengan akun Github yang sudah dibuat pada langkah-langkah sebelumnya. Ikuti langkah-langkah baris perintah berikut ini:

a. Autentikasi akun

```
git config --global user.name "suendri-uinsu"
git config --global user.email "suendri@uinsu.ac.id"
```



```
MINGW64/c/xampp/htdocs/oop-praktikum-8
suendri@DESKTOP-F0CB97K MINGW64 /c/xampp/htdocs/oop-praktikum-8
$ git config --global user.name "suendri-uinsu"
suendri@DESKTOP-F0CB97K MINGW64 /c/xampp/htdocs/oop-praktikum-8
$ git config --global user.email "suendri@uinsu.ac.id"
suendri@DESKTOP-F0CB97K MINGW64 /c/xampp/htdocs/oop-praktikum-8
$ git config --list
```

b. Inisialisasi, untuk memperkenalkan *folder* tersebut pada git.

```
git init
```

c. Menambahkan semua isi folder **oop-praktikum-8** ke Git.

```
git add --all
```

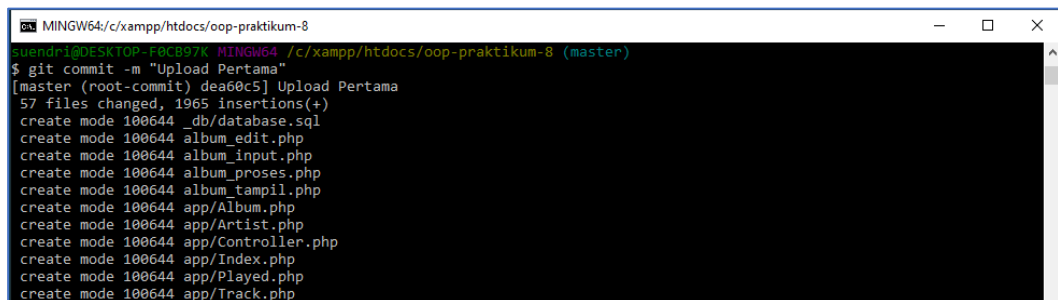
Jika anda sudah mempunyai *composer* pada project tersebut, pastikan Inisialisasi pada *composer* sudah sesuai dengan repositori yang akan dituju.

d. Commit, berfungsi untuk menyimpan perubahan

```
Git commit -m "Komentar perubahan"
```

Contoh:

```
Git commit -m "Upload pertama"
```



```
MINGW64/c/xampp/htdocs/oop-praktikum-8
suendri@DESKTOP-F0CB97K MINGW64 /c/xampp/htdocs/oop-praktikum-8 (master)
$ git commit -m "Upload Pertama"
[master (root-commit) dea60c5] Upload Pertama
 57 files changed, 1965 insertions(+)
 create mode 100644 db/database.sql
 create mode 100644 album_edit.php
 create mode 100644 album_input.php
 create mode 100644 album_proses.php
 create mode 100644 album_tampil.php
 create mode 100644 app/Album.php
 create mode 100644 app/Artist.php
 create mode 100644 app/Controller.php
 create mode 100644 app/Index.php
 create mode 100644 app/Played.php
 create mode 100644 app/Track.php
```

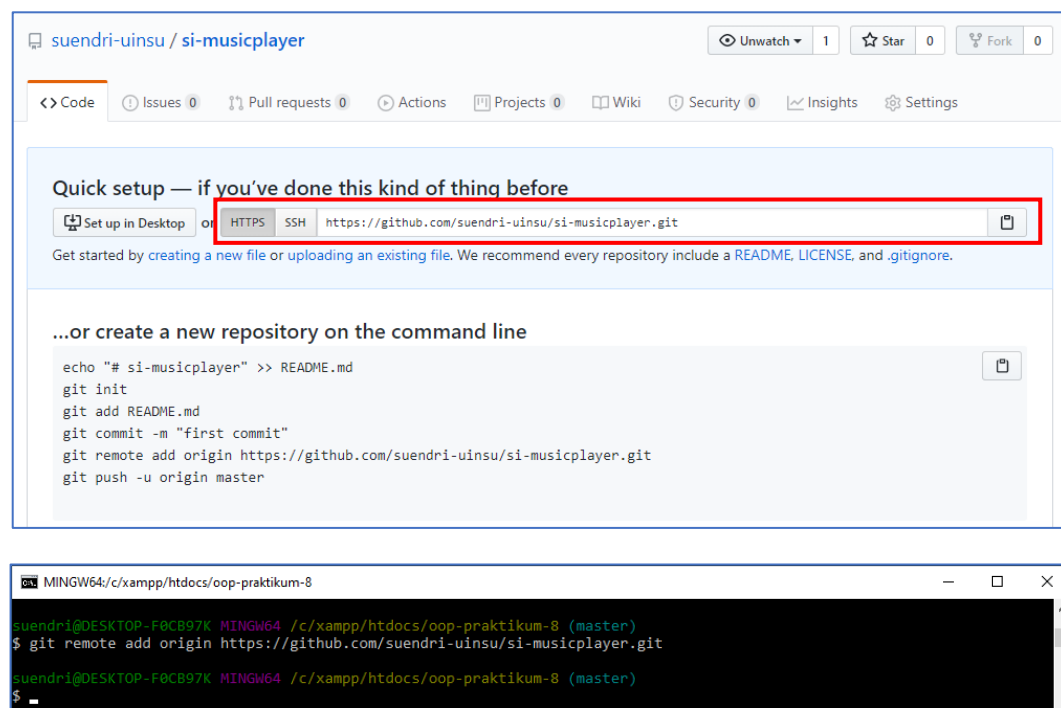

- e. Remote, perintah ini berfungsi untuk menghubungkan ke akun Github anda.

```
git remote add origin alamat.git
```

Contoh:

```
git remote add origin https://github.com/suendri-uinsu/si-musicplayer.git
```

Alamat git anda dapatkan dari repositori yang sudah anda buat sebelumnya, buka repositori dan copy alamat git.



- f. Push, mengirim project ke Github.

```
git push -u origin master
```

Masukkan username dan password yang diminta, semua file dalam project tersebut akan diupload ke Github.

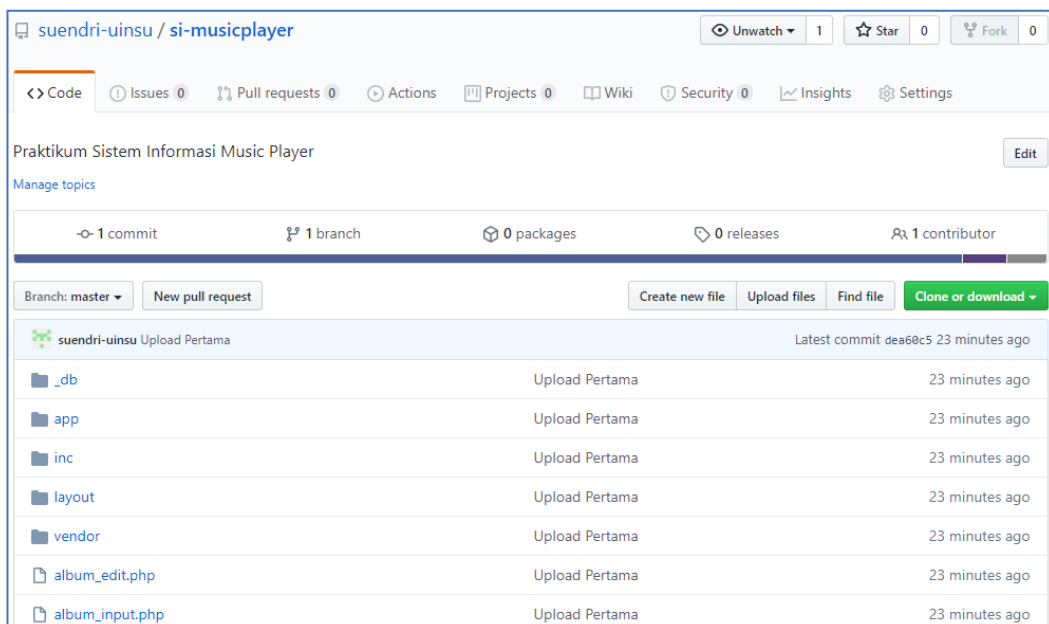
```
MINGW64/c/xampp/htdocs/oop-praktikum-8
suendri@DESKTOP-F0CB97K MINGW64 /c/xampp/htdocs/oop-praktikum-8 (master)
$ git remote add origin https://github.com/suendri-uinsu/si-musicplayer.git

suendri@DESKTOP-F0CB97K MINGW64 /c/xampp/htdocs/oop-praktikum-8 (master)
$ git push -u origin master
Enumerating objects: 66, done.
Counting objects: 100% (66/66), done.
Delta compression using up to 2 threads
Compressing objects: 100% (61/61), done.
Writing objects: 100% (66/66), 7.75 MiB | 661.00 KiB/s, done.
Total 66 (delta 10), reused 0 (delta 0)
remote: Resolving deltas: 100% (10/10), done.
To https://github.com/suendri-uinsu/si-musicplayer.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

suendri@DESKTOP-F0CB97K MINGW64 /c/xampp/htdocs/oop-praktikum-8 (master)
$
```

Upload project ke repository Github sudah selesai, kunjungi alamat repository.

<https://github.com/suendri-uinsu/si-musicplayer>



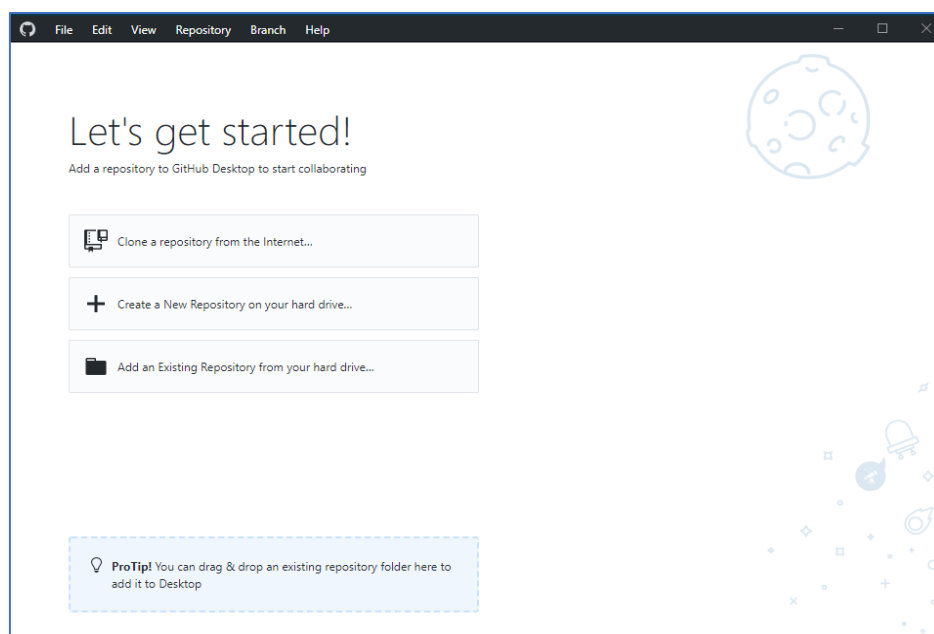
Masih banyak perintah lainnya yang bisa anda gunakan, silakan pelajari dan praktikkan.

2. Menggunakan Aplikasi Desktop

Aplikasi desktop telah disediakan oleh Github, tentunya ini yang paling mudah tanpa menghafal dan mengetikkan kode. Aplikasi tersebut dapat anda temukan di alamat <https://desktop.github.com/>. Download aplikasi tersebut dan Install. Instalasi hanya 1 langkah saja dan langsung terpasang di komputer anda.



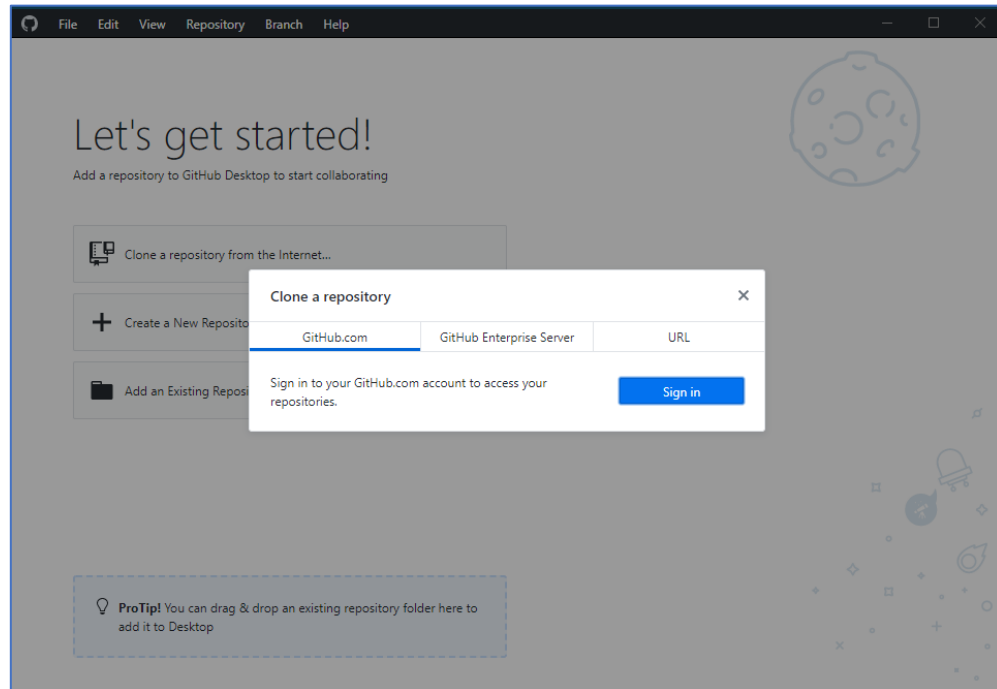
Berikut ini merupakan tampilan Github Desktop dengan tampilan yang mudah untuk digunakan.



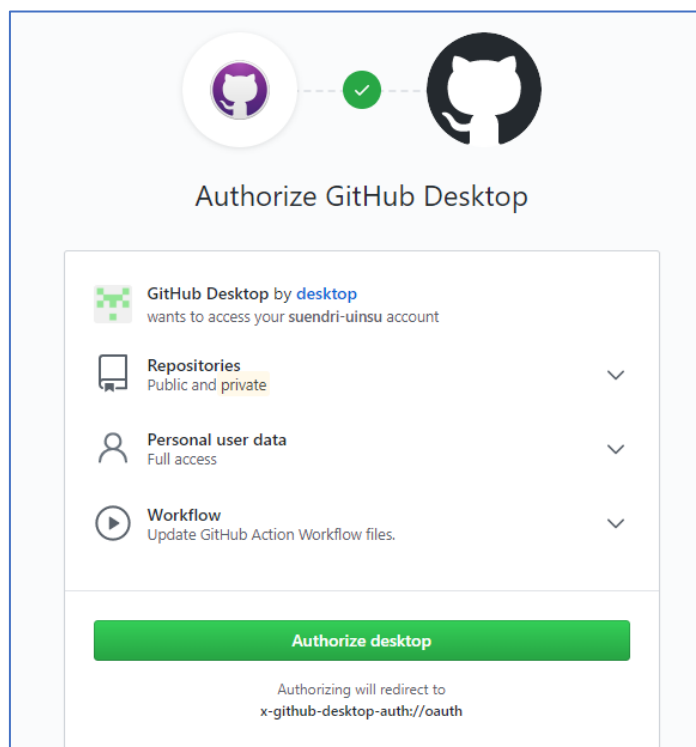
- Clone**, berfungsi untuk mengkloning project yang sudah ada di Github ke hardrive.
- Create a New**, berfungsi untuk membuat repositori baru
- Add an Existing**, berfungsi untuk menambah repositori yang sudah ada di hardrive

Suendri - Modul Praktikum Pemrograman Berbasis Lanjutan

Karena repositori sebelumnya sudah ada, kita coba untuk memilih opsi **Clone**. Tahap ini bisa dilakukan dengan syarat belum ada nama folder yang sama di c:xampp/htdocs.

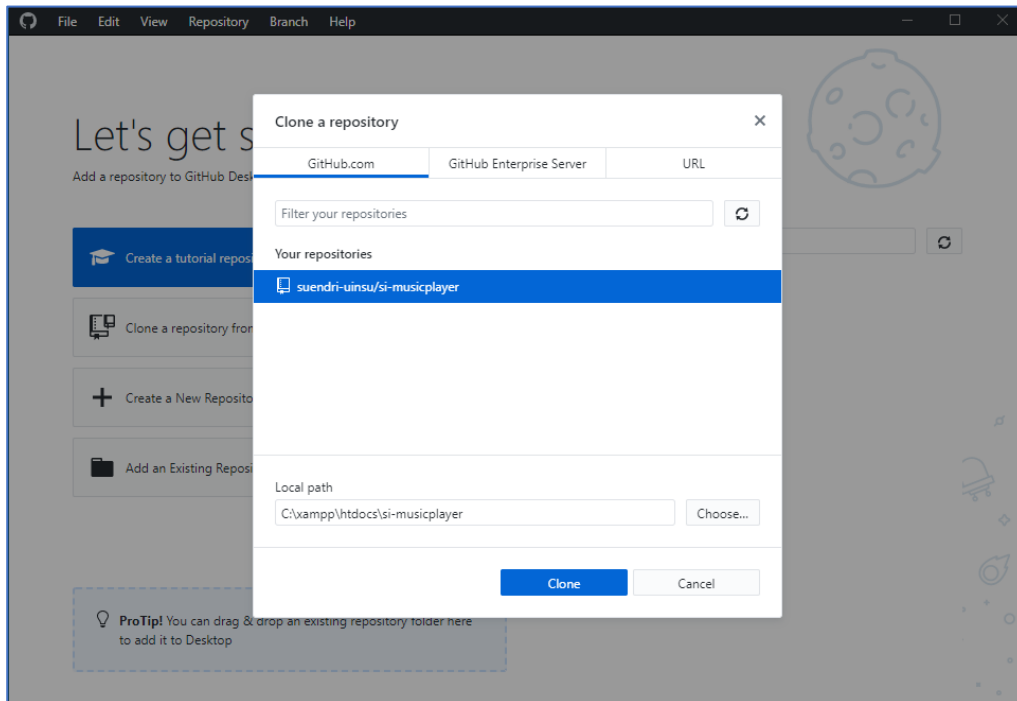


Login menggunakan akun yang sudah ada. Jika berhasil, Github akan meminta otorisasi melalui browser, Izinkan otorisasi.

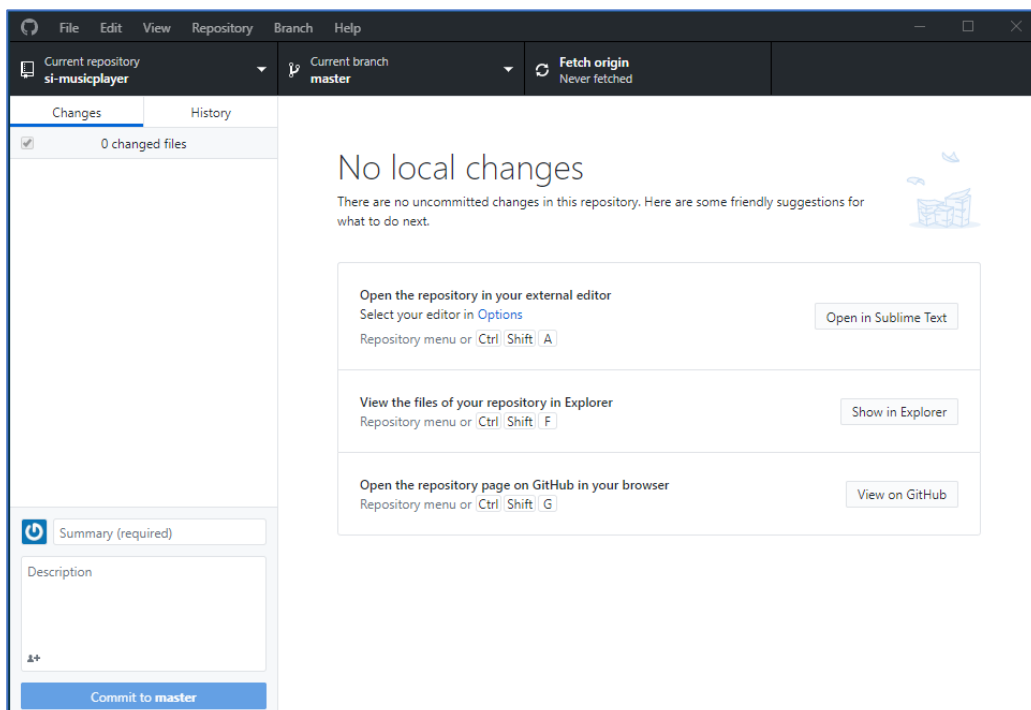


Suendri - Modul Praktikum Pemrograman Berbasis Lanjutan

Clone repositori **si-musicplayer** dan pilih *folder* c:xampp/htdocs atau *folder* lain yang anda inginkan.



Seluruh perubahan yang terjadi akan tampil di kolom bagian kiri, untuk menyimpan perubahan klik **Commit** dan untuk *upload* ke Github klik **Push**.



8.3 Latihan

Lengkapi project Ujian Tengah Semester masing-masing yang telah anda rancang, kemudian sesuaikan dengan struktur project si-musicplayer pada praktikum ini. Upload project masing-masing ke Github.

MODUL 9

MVC

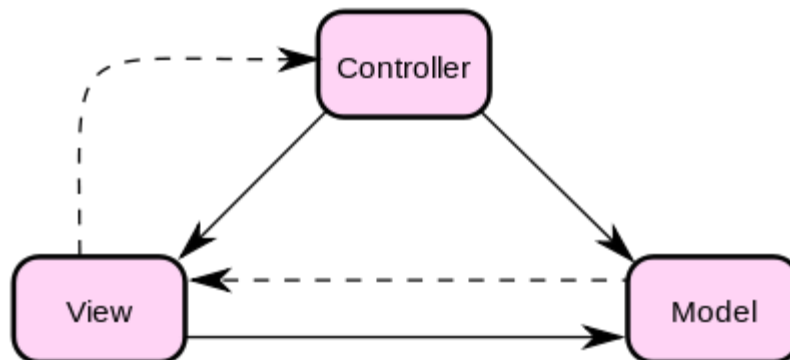
9.1 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

- Memahami teknik *Model View Controller* (MVC) menggunakan bahasa pemrograman PHP.
- Merancang program menggunakan teknik *Model View Controller* (MVC).

9.2 Praktikum

Model View Controller atau MVC adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (*Model*) dari tampilan (*View*) dan cara bagaimana memprosesnya (*Controller*). Dalam implementasinya kebanyakan kerangka kerja (*framework*) dalam aplikasi web adalah berbasis arsitektur MVC. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, antarmuka pengguna, dan bagian yang menjadi kontrol dalam sebuah aplikasi web (Wikipedia).



Seperti namanya, ada tiga komponen yang dapat dideskripsikan sebagai berikut:

- Model* berfungsi untuk mengatur data, fungsi dan aturan dari aplikasi.
- View* berfungsi untuk mengatur tampilan atau *output* yang tampil di layar, tidak hanya berupa data, namun juga termasuk komponen lain, seperti gambar, video, diagram, dan sebagainya.

3. *Controller* merupakan program yang mengatur, menerima input dan menjalankan beberapa perintah untuk dijalankan di model.

Design pattern MVC ini dikembangkan dengan tujuan untuk membuat sebuah program yang dapat digunakan secara berulang untuk hal yang serupa, dan dikembangkan dengan modul tambahan sehingga tidak terjadi proses pengulangan pengembangan dari nol. *Don't reinvent the wheel* – pepatah yang seringkali didengungkan di dunia pengembangan aplikasi, merupakan salah satu tujuan utama pemanfaatan teknik *Model View Controller*.

Dengan pemisahan seperti ini, kerja tim menjadi mudah dikelola. Selain itu dengan penerapan konsep MVC yang baik, setiap bagian tidak saling bergantung sama lain. Jika ada perubahan atau modifikasi, cukup edit di bagian yang diperlukan saja, tidak harus merombak ulang semua aplikasi. Namun dibalik keunggulan ini, kendala utama dari konsep MVC adalah cukup rumit untuk dipahami (terutama bagi pemula), serta file kode program menjadi banyak karena setiap bagian dari MVC harus ditulis dalam file terpisah. Namun keuntungan yang didapat sebanding dengan “usaha” untuk mempelajari MVC tersebut, karena kode program kita menjadi lebih fleksibilitas dan mudah dikelola

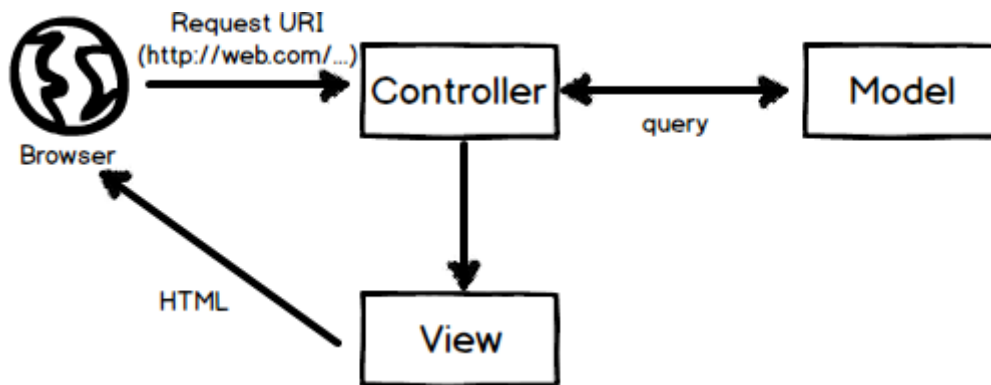
Merancang program MVC

Program MVC bisa dibangun sendiri atau menggunakan program yang sudah dibangun oleh komunitas tertentu. Banyak kerangka kerja yang sudah dibangun oleh *programmer* yang tentunya sudah sangat ahli pada bahasa pemrograman tersebut, dengan keunggulan-keunggulan dan kemudahan yang ditawarkan, kita bisa menggunakan *framework* tersebut sesuai dengan kebutuhan. Banyak *framework* yang dibangun menggunakan teknik MVC diantaranya;

1. Laravel
2. Symfony
3. CodeIgniter
4. CakePHP
5. Yii

Framework tersebut diatas merupakan *framework* yang dibangun menggunakan bahasa pemrograman PHP sebagaimana bahasa pemrograman yang digunakan pada

modul ini, adapun *framework Laravel* akan dibahas secara khusus pada modul berikutnya.



Persiapan

1. *Composer*, pastikan komputer anda sudah tersedia aplikasi *composer* sebagaimana yang telah dibahas pada modul 7 sebelumnya.
2. Koneksi Internet, karena kita menggunakan *composer*, anda wajib terkoneksi dengan internet.
3. Buat folder baru di **c:xampp/htdocs/oop-praktikum-9**
4. Jalankan inisialisasi *Composer* dari *Command Prompt*

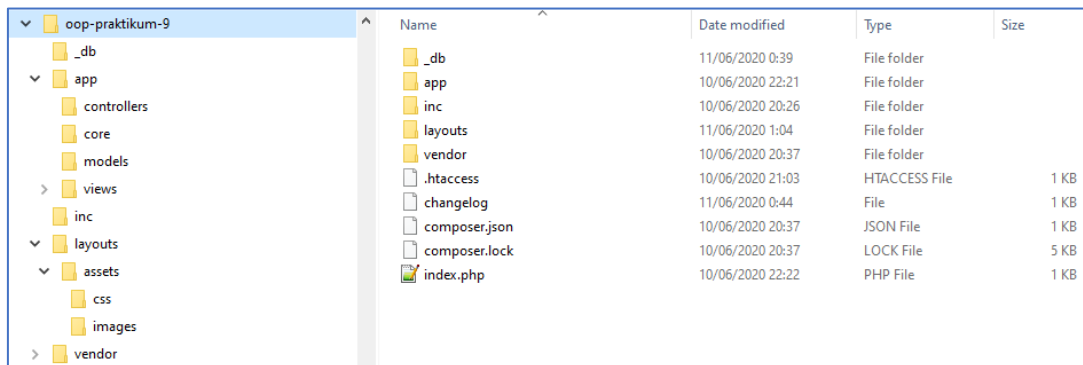
composer init

sesuaikan dengan contoh berikut ini, gunakan profil akun Github anda.

```
1. {
2.     "name": "suendri-uinsu/oop-praktikum-9",
3.     "description": "Praktikum 9 Pemrograman Berbasis Web Lanjutan",
4.     "type": "project",
5.     "authors": [
6.         {
7.             "name": "suendri-uinsu",
8.             "email": "suendri@uinsu.ac.id"
9.         }
10.    ],
11.    "require": {
12.        "filp/whoops": "^2.7"
13.    },
14.    "autoload": {
15.        "psr-4": {
16.            "App\\": "app/"
17.        }
18.    }
19. }
```

5. Desain struktur *folder* dan *file* seperti berikut ini.

```
1. | _db
2. |   database.sql
3. | app
4. |   controllers
5. |     Dashboard.php
6. |     Index.php
7. |     User.php
8. |   core
9. |     Bootstrap.php
10. |    Controller.php
11. |    Error.php
12. |    Model.php
13. |   models
14. |   views
15. | inc
16. |   config.php
17. | layouts
18. |   assets
19. |     css
20. |       style.css
21. |   dashboard.php
22. |   index.php
23. | .htaccess
24. | index.php
```



The screenshot shows a file explorer window with the directory structure of 'oop-praktikum-9'. The left pane shows a tree view with folders expanded: _db, app, controllers, core, models, views, inc, layouts, assets, css, images, and vendor. The right pane shows a list of files and folders with columns for Name, Date modified, Type, and Size.

Name	Date modified	Type	Size
_db	11/06/2020 0:39	File folder	
app	10/06/2020 22:21	File folder	
inc	10/06/2020 20:26	File folder	
layouts	11/06/2020 1:04	File folder	
vendor	10/06/2020 20:37	File folder	
.htaccess	10/06/2020 21:03	HTACCESS File	1 KB
changelog	11/06/2020 0:44	File	1 KB
composer.json	10/06/2020 20:37	JSON File	1 KB
composer.lock	10/06/2020 20:37	LOCK File	5 KB
index.php	10/06/2020 22:22	PHP File	1 KB

6. Tambahkan *PSR-4 Autoload* pada file **composer.json**

```
1. "autoload": {
2.     "psr-4": {
3.         "App\\": "app/"
4.     }
5. }
```

7. Untuk memudahkan *monitoring error*, anda bisa memasang salah satu paket yang sudah tersedia di Github seperti *filp/whoops*. *Install* menggunakan perintah berikut ini.

```
composer require filp/whoops
```

8. Tambahkan baris perintah berikut ini **inc/config.php**

```
1. <?php
2.
3. // Laporan error
4. error_reporting(E_ALL);
5.
6. // Mulai sesi
7. session_start();
8.
9. // Url -
10. define("URL", "http://localhost/oop-praktikum-9");
11. define("AST", URL . "/layouts/assets");
12.
13. // Path root
14. define("ROOT", dirname(__DIR__) . DIRECTORY_SEPARATOR);
15.
16. // Autoload
17. require_once ROOT . "vendor/autoload.php";
18.
19. // Whoops
20. $whoops = new \Whoops\Run;
21. $whoops->pushHandler(new \Whoops\Handler\PrettyPageHandler);
22. $whoops->register();
```

9. Tambahkan baris perintah berikut di **app/core/Bootstrap.php**

```
1. <?php
2.
3. namespace App\Core;
4.
5. /**
6.  * controller/action/params
7.  */
8. class Bootstrap
9. {
10.
11.     public function __construct()
12.     {
13.         // Jika url diakses
14.         if (isset($_GET['page'])) {
15.             // filter_var = Filter Url
16.             $url = filter_var($_GET['page'], FILTER_SANITIZE_URL);
17.             // trim = Hilangkan space
18.             $url = trim($url);
19.             // explode = Membagi string diantara slash
```

```
20.         $url = explode('/', $url);
21.         // ucfirst = Uppercase First
22.         // array_shift = mengambil nilai pertama array
23.         $page = ucfirst(array_shift($url));
24.
25.         if (file_exists(ROOT . "app/controllers/" . $page . ".php")) {
26.             $class = "App\\Controllers\\" . $page;
27.             $controller = new $class;
28.             // cek method
29.             $action = array_shift($url);
30.             if (method_exists($controller, $action)) {
31.                 // Parameters = controller/detail/1
32.                 $params = array_values($url);
33.                 if(!empty($params)) {
34.                     call_user_func_array(array($controller, $action), $params);
35.                 } else {
36.                     $controller->{$action}(@$url);
37.                 }
38.             } else {
39.                 $controller->index();
40.             }
41.         } else {
42.             $class = "App\\Core\\Error";
43.             $controller = new $class();
44.             $controller->fileNotFound();
45.         }
46.
47.     } else {
48.         $class = "App\\Controllers\\Index";
49.         $controller = new $class();
50.         $controller->index();
51.     }
52. }
53. }
```

10. Tambahkan baris perintah berikut di **.htaccess**

```
1. RewriteEngine on
2. RewriteCond %{REQUEST_FILENAME} !-f
3. RewriteCond %{REQUEST_FILENAME} !-d
4. RewriteRule ^(.*)$ index.php?page=$1 [NC,L,QSA]
```

11. Tambahkan baris perintah berikut di index.php

```
1. <?php
2.
3. require_once "inc/config.php";
4.
5. new App\\Core\\Bootstrap();
```

12. Sampai disini, pondasi MVC sudah tersedia. Langkah selanjutnya anda akan membangun tata letak *Model View* dan *Controller*.

9.3 Latihan

Kode program praktikum tersebut diatas sudah tersedia pada akun github dosen pengampu di <https://github.com/suendri-uinsu/si-mvc>. Analisis, Sempurnakan dan Buatlah tampilan Create Read Update dan Delete (CRUD) data user menggunakan Design Patter MVC tersebut.

MODUL 10

PHP FRAMEWORK

10.1 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

- a. Memahami dan mengetahui jenis-jenis Framework menggunakan bahasa pemrograman PHP.
- b. Memahami dan mengetahui langkah-langkah instalasi Framework Laravel.

10.2 Praktikum

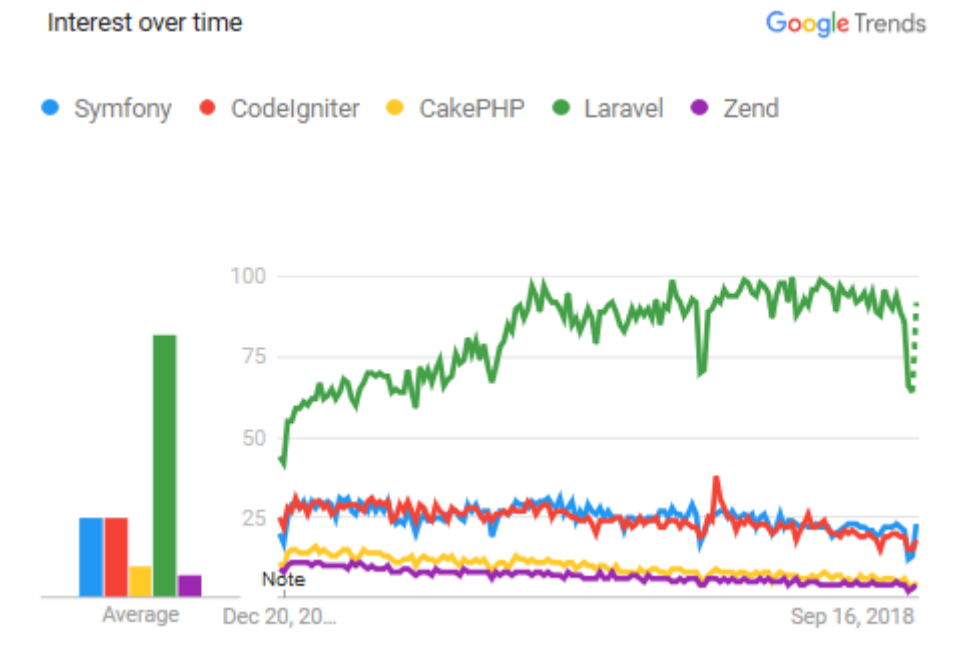
Framework atau bahasa indonesianya kerangka kerja adalah sekumpulan susunan kode program untuk memudahkan para *programmer* membuat aplikasi atau web yang isinya adalah berbagai fungsi, plugin, paket dan konsep sehingga membentuk suatu sistem tertentu. Dengan menggunakan framework, sebuah aplikasi akan tersusun dan terstruktur dengan rapi mengikuti urutan yang telah ditentukan oleh pengembang *framework* tersebut. Framework pertama kali yang digunakan para *programmer* muncul pada tahun 2004 yaitu Prado 1. Kemudian seiring berjalannya waktu dan ilmu komputer yang semakin lama semakin berkembang, muncul beberapa framework baru yang digunakan oleh para *programmer* seperti CakePHP, Symfony, CodeIgniter, Zend Framework, Kohana, Yii, dan yang paling banyak digunakan para programmer sekarang adalah framework Laravel.

Kelebihan *framework* antara lain sebagai berikut :

- a. Ringan dan cepat. Framework hanya melakukan pemanggilan pustaka/kelas yang dibutuhkan sehingga meminimalkan *resource* yang diperlukan sehingga ketika kita me-load sebuah halaman akan menjadi ringan dan cepat.
- b. Menggunakan metode MVC. Seperti yang telah dijelaskan sebelumnya, dengan metode MVC akan mempermudah kita dalam memahami alur pemrograman karena untuk bagian tampilan, logika dan query database telah dipecah sedemikian rupa.
- c. Mayoritas mendukung berbagai jenis database.
- d. Tingkat keamanan yang tinggi dan terbaru.

PHP Framework Trend

Berdasarkan survey yang dilakukan oleh coderseye.com pada tahun 2020 yang terdapat pada website <https://coderseye.com/best-php-frameworks-for-web-developers/>. Bahwa ada 11 PHP Framework terbaik yang paling sering digunakan atau istilah lain paling trend dikalangan programmer. Grafik trend yang dikumpulkan oleh google diantaranya bisa dilihat pada grafik berikut ini



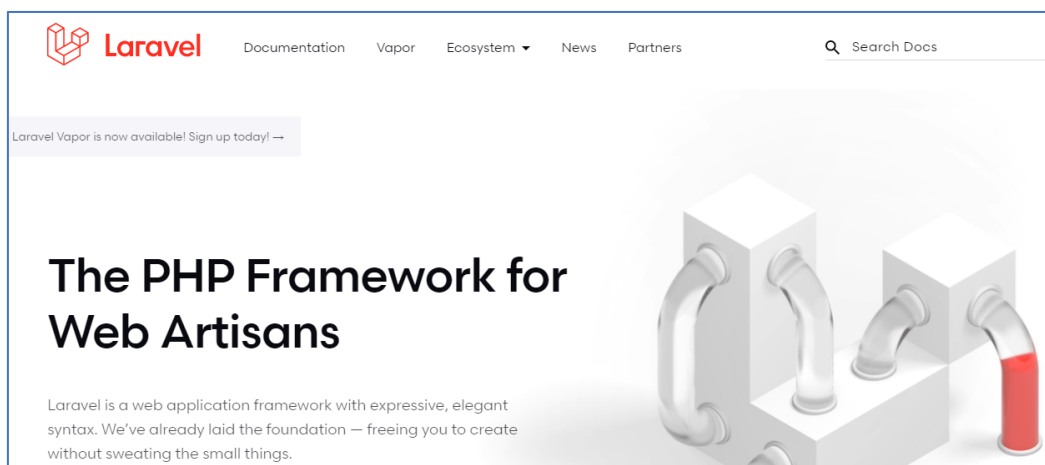
Adapun 11 PHP Framework yang berhasil disurvei oleh coderseye.com pada tahun 2020 adalah sebagai berikut:

1. Laravel
2. Phalcon
3. Codeigniter
4. Symphony
5. CakePHP
6. Zend Framework
7. FuelPHP
8. Slim
9. Phpixie
10. Fat-free
11. Aura

Laravel

Laravel diluncurkan sejak tahun 2011 dan mengalami pertumbuhan yang cukup eksponensial. Di tahun 2015, Laravel adalah framework yang paling banyak mendapatkan bintang di Github. Sekarang framework ini menjadi salah satu yang populer di dunia, tidak terkecuali di Indonesia. Laravel fokus di bagian end-user, yang berarti fokus pada kejelasan dan kesederhanaan, baik penulisan maupun tampilan, serta menghasilkan fungsionalitas aplikasi web yang bekerja sebagaimana mestinya. Hal ini membuat developer maupun perusahaan menggunakan framework ini untuk membangun apa pun, mulai dari proyek kecil hingga skala perusahaan kelas atas.

Laravel mengubah pengembangan website menjadi lebih elegan, ekspresif, dan menyenangkan, sesuai dengan jargonnya “The PHP Framework For Web Artisans”. Selain itu, Laravel juga mempermudah proses pengembangan website dengan bantuan beberapa fitur unggulan, seperti Template Engine, Routing, dan Modularity. Framework Laravel bisa ditemukan di web resminya laravel.com, dengan dokumentasi yang lengkap, tutorial-tutorial yang mudah untuk dipahami, baik dalam bentuk text maupun video serta dukungan komunitas yang besar, membuat Framework ini mudah untuk dipelajari dan digunakan.



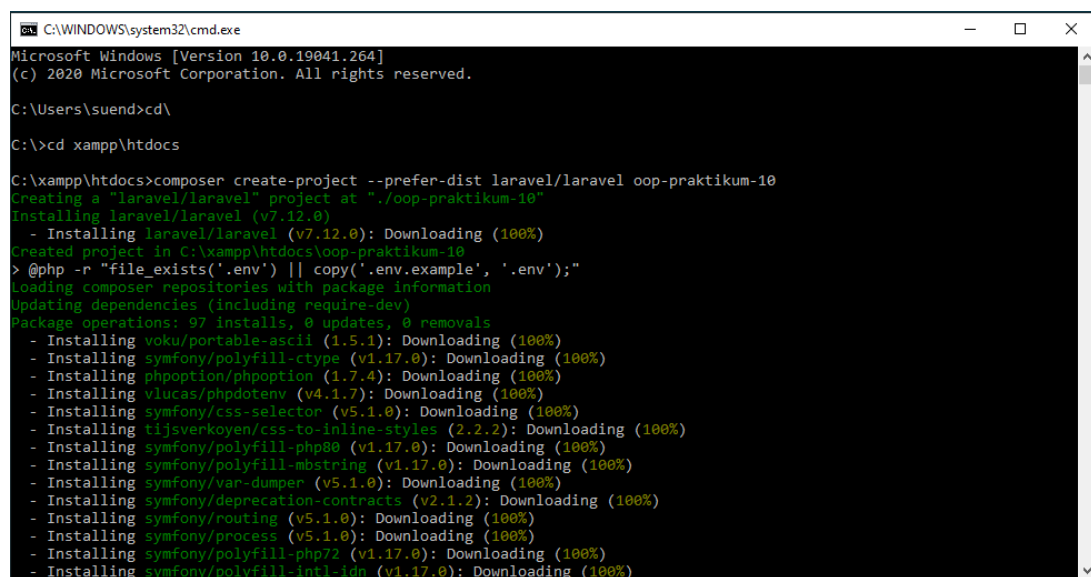
Instalasi

Saat modul ini di buat, laravel sudah merilis ver 7.15, adapun versi 7.x mempunyai langkah yang sama. Ikuti langkah-langkah berikut untuk memasang framework laravel di komputer anda masing-masing. Langkah instalasi ini juga bisa anda temukan di dokumentasi Laravel di <https://laravel.com/docs/7.x>

1. PHP >= 7.2.5, pada komputer anda sudah terinstall PHP 7.4.6, tentunya ini sudah mendukung versi yang direkomendasikan.
2. *Composer*, pastikan komputer anda sudah tersedia aplikasi *composer* sebagaimana yang telah dibahas pada modul 7 sebelumnya.
3. Koneksi Internet, karena kita menggunakan *composer*, anda wajib terkoneksi dengan internet.
4. Gunakan *Command Prompt* untuk menjalankan baris perintah yang disediakan.
5. Jalankan perintah berikut ini di **c:/xampp/htdocs**

```
composer create-project --prefer-dist laravel/laravel oop-praktikum-10
```

composer akan memasang seluruh paket yang dibutuhkan lebih kurang 37 MB.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19041.264]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\suend>cd\
C:\>cd xampp\htdocs

C:\xampp\htdocs>composer create-project --prefer-dist laravel/laravel oop-praktikum-10
Creating a "laravel/laravel" project at "../oop-praktikum-10"
Installing laravel/laravel (v7.12.0)
- Installing laravel/laravel (v7.12.0): Downloading (100%)
Created project in C:\xampp\htdocs\oop-praktikum-10
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 97 installs, 0 updates, 0 removals
- Installing voku/portable-ascii (1.5.1): Downloading (100%)
- Installing symfony/polyfill-ctype (v1.17.0): Downloading (100%)
- Installing phoption/phoption (1.7.4): Downloading (100%)
- Installing vlucas/phpdotenv (v4.1.7): Downloading (100%)
- Installing symfony/css-selector (v5.1.0): Downloading (100%)
- Installing tijsverkoyen/css-to-inline-styles (2.2.2): Downloading (100%)
- Installing symfony/polyfill-php80 (v1.17.0): Downloading (100%)
- Installing symfony/polyfill-mbstring (v1.17.0): Downloading (100%)
- Installing symfony/var-dumper (v5.1.0): Downloading (100%)
- Installing symfony/deprecation-contracts (v2.1.2): Downloading (100%)
- Installing symfony/routing (v5.1.0): Downloading (100%)
- Installing symfony/process (v5.1.0): Downloading (100%)
- Installing symfony/polyfill-php72 (v1.17.0): Downloading (100%)
- Installing symfony/polyfill-intl-idn (v1.17.0): Downloading (100%)
```

6. Buka *browser* anda dan masukkan ke URL <http://localhost/oop-praktikum-10/public/>



7. Jika *halaman* utama Laravel sudah tampil seperti gambar tersebut diatas, berarti Laravel sudah berhasil diunduh dan dipasang di komputer anda. Selanjutnya, buka **phpMyAdmin** dan buat database dengan nama :

dbpraktikum10

8. Bukalah *folder* **oop-praktikum-10** dari text editor dan setting koneksi.

buka file **.env** menggunakan editor dan atur seperti kode perintah berikut ini (warna biru):

```
APP_NAME="Praktikum 10 PBWL UINSU - PHP Framework"
APP_ENV=local
APP_KEY=base64:v5xJh+Hg208C64r37ixb/aiP4edtLkc0Dyrh2wipVBw=
APP_DEBUG=true
APP_URL=http://localhost/oop-praktikum-10/public

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=dbpraktikum10
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
...
```

9. Kembali ke *Command Prompt* dan masuk ke direktori **oop-praktikum-10**

cd oop-praktikum-10

10. Laravel sudah menyediakan *Form Login*, anda bisa dengan mudah memasangnya pada *framework* yang baru diinstal.

```
composer require laravel/ui
php artisan ui bootstrap --auth
```

```
C:\WINDOWS\system32\cmd.exe - composer require laravel/ui

C:\xampp\htdocs\oop-praktikum-10>composer require laravel/ui
Using version ^2.0 for laravel/ui
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Installing laravel/ui (v2.0.3): Loading from cache
Writing lock file
Generating optimized autoload files
```

11. Selanjutnya, Laravel juga menyediakan perintah untuk mengatur tabel-tabel dalam database yang disebut dengan *Migration*. Gunakan perintah berikut untuk membuat tabel user untuk menyimpan data login secara otomatis.

php artisan migrate

```
C:\WINDOWS\system32\cmd.exe

C:\xampp\htdocs\oop-praktikum-10>php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table (0.17 seconds)
Migrated: 2014_10_12_000000_create_users_table (0.17 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table (0.3 seconds)
Migrated: 2014_10_12_100000_create_password_resets_table (0.3 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table (0.08 seconds)
Migrated: 2019_08_19_000000_create_failed_jobs_table (0.08 seconds)
C:\xampp\htdocs\oop-praktikum-10>
```

Database akan terisi otomatis, anda bisa lihat dari phpMyAdmin

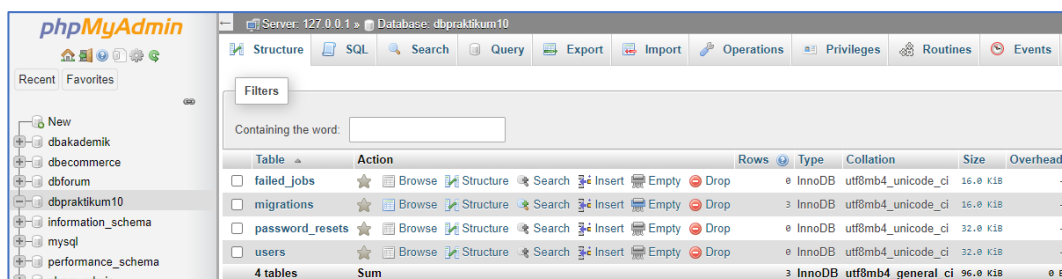


Table	Action	Rows	Type	Collation	Size	Overhead
failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 K1B	-
migrations	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16.0 K1B	-
password_resets	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 K1B	-
users	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 K1B	-
4 tables	Sum	3	InnoDB	utf8mb4_general_ci	96.0 K1B	0 B

12. Terakhir, buka kembali browser anda dan *refresh*, pastikan Menu Login dan Register sudah ada di bagian kanan atas Framework Laravel anda.



Klik menu *Register* untuk membuat akun baru dan *Login* untuk masuk ke dalam Sistem sesuai dengan akun yang anda registrasikan.

10.3 Latihan

Ikuti langkah-langkah instalasi tersebut diatas dan praktikan pada komputer masing-masing. Ulangi langkah tersebut beberapa kali agar mudah untuk diingat.

10.4 Praktikum View

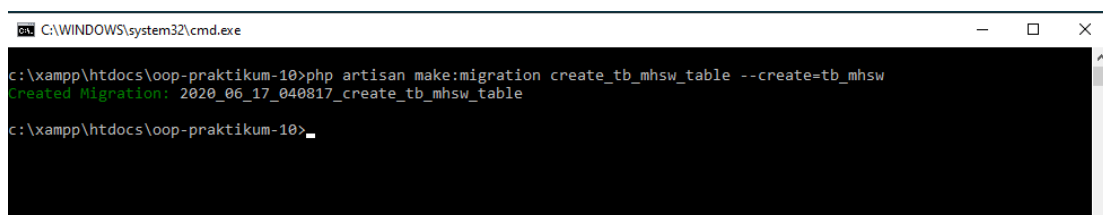
Ada dua cara untuk membuat tabel-tabel pada praktikum ini :

- Menggunakan phpMyAdmin
- Menggunakan *Migration*

Karena sudah menggunakan *Framework* Laravel, kita akan coba membuat tabel-tabel yang digunakan pada praktikum ini dengan bantuan fitur Migration.

- Buka kembali *Command Prompt* anda, pastikan sudah ada di folder **c:/xampp/htdocs/oop-praktikum-10**. Jalankan kode perintah berikut ini:

```
php artisan make:migration create_tb_mhsw_table --create=tb_mhsw
```



```
C:\WINDOWS\system32\cmd.exe
c:\xampp\htdocs\oop-praktikum-10>php artisan make:migration create_tb_mhsw_table --create=tb_mhsw
Created Migration: 2020_06_17_040817_create_tb_mhsw_table
c:\xampp\htdocs\oop-praktikum-10>
```

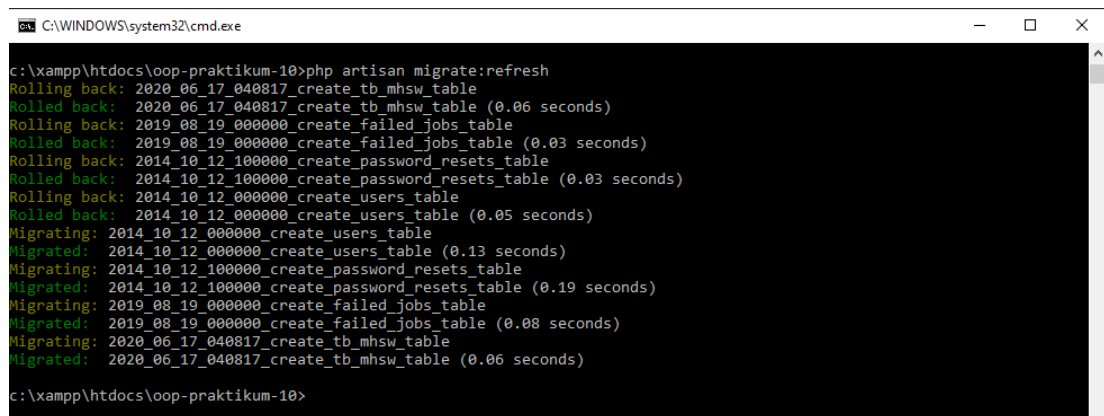
Perintah tersebut diatas akan membuat sebuah *file* baru di **app/database/migrations/date_create_table.php** dan isikan baris perintah berikut :

```
1. public function up()
2. {
3.     Schema::create('tb_mhsw', function (Blueprint $table) {
4.         $table->increments('mhsw_id');
5.         $table->string('mhsw_nim');
6.         $table->string('mhsw_nama');
7.         $table->string('mhsw_jurusan');
```

```
8.         $table->string('mhsw_alamat');
9.         $table->timestamps();
10.    });
11.    }
```

Kemudian jalankan perintah berikut untuk eksekusi ke Database.

```
php artisan migrate:refresh
```



```
C:\WINDOWS\system32\cmd.exe
c:\xampp\htdocs\oop-praktikum-10>php artisan migrate:refresh
Rolling back: 2020_06_17_040817_create_tb_mhsw_table
Rolling back: 2020_06_17_040817_create_tb_mhsw_table (0.06 seconds)
Rolling back: 2019_08_19_000000_create_failed_jobs_table
Rolling back: 2019_08_19_000000_create_failed_jobs_table (0.03 seconds)
Rolling back: 2014_10_12_100000_create_password_resets_table
Rolling back: 2014_10_12_100000_create_password_resets_table (0.03 seconds)
Rolling back: 2014_10_12_000000_create_users_table
Rolling back: 2014_10_12_000000_create_users_table (0.05 seconds)
Migrating: 2014_10_12_000000_create_users_table
Migrating: 2014_10_12_000000_create_users_table (0.13 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrating: 2014_10_12_100000_create_password_resets_table (0.19 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrating: 2019_08_19_000000_create_failed_jobs_table (0.08 seconds)
Migrating: 2020_06_17_040817_create_tb_mhsw_table
Migrating: 2020_06_17_040817_create_tb_mhsw_table (0.06 seconds)
c:\xampp\htdocs\oop-praktikum-10>
```

2. Buat **Model** untuk tabel **tb_mhsw** dengan perintah:

```
php artisan make:model Mahasiswa
```

Perintah diatas akan membuat sebuah file baru di **app/Mahasiswa.php**, atur baris perintah di **Mahasiswa.php** seperti pada berikut ini.

```
1. class Mahasiswa extends Model
2. {
3.     protected $table = "tb_mhsw";
4.
5.     protected $primaryKey = 'mhsw_id';
6.
7.     protected $fillable = ['mhsw_nim', 'mhsw_nama',
8.         'mhsw_jurusan', 'mhsw_alamat'];
```

3. Buat **Controller** baru untuk **tb_mhsw** dengan perintah berikut ini:

```
php artisan make:controller MahasiswaController --resource
```

Perintah diatas akan membuat sebuah *file* baru di **app/Http/Controllers/MahasiswaController.php**, atur baris perintah di **MahasiswaController.php** seperti pada berikut ini:

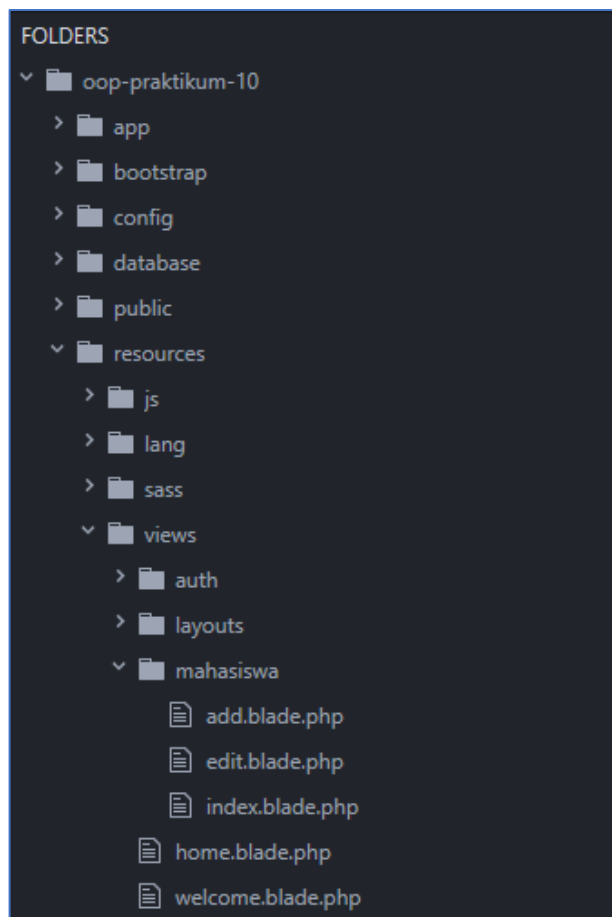
```
1. <?php
2.
3. namespace App\Http\Controllers;
4.
5. use Illuminate\Http\Request;
6. use App\Mahasiswa;
7.
8. class MahasiswaController extends Controller
9. {
10.
11.     public function index()
12.     {
13.         $rows = Mahasiswa::all();
14.         return view('mahasiswa.index', compact('rows'));
15.     }
16.
17.     public function create()
18.     {
19.         return view('mahasiswa.add');
20.     }
}
```

Baris perintah ke-6 merupakan perintah untuk memanggil Model **app/Mahasiswa.php**

Method **index()** untuk memanggil View dari **resources/views/mahasiswa/index.blade.php** yang akan kita buat pada langkah berikutnya.

Method **create()** untuk memanggil View dari **resources/views/mahasiswa/add.blade.php** yang akan kita buat pada langkah berikutnya.

4. Buat **View** baru dengan cara membuat direktori baru di **resources/views** dengan nama **mahasiswa**. Buat *file* baru sekaligus dengan nama **index.blade.php**, **add.blade.php**, **edit.blade.php**, **detail.blade.php**



Blade merupakan ekstensi *template* yang disediakan oleh Laravel. Setiap *file* yang digunakan untuk *view* menggunakan ekstensi yang disediakan tersebut dengan format **namaview.blade.php**

Controller akan memanggil *View* dengan perintah

view(namafolder.namafile)

seperti yang terdapat pada *method* **index()** dan **create()** pada *file controller* **MahasiswaController.php**

5. Selanjutnya, atur *View* menggunakan *template default* yang sudah tersedia sebelumnya di **resources/views/layouts/app.blade.php**, template ini merupakan bawaan Laravel, anda bisa mengubah template tersebut sesuai kebutuhan. Buka file **mahasiswa/index.blade.php**, isikan contoh view berikut

```
1. @extends('layouts.app')
2.
3. @section('content')
4.
5. <div class="container">
6.
7.     <h3>Daftar Mahasiswa</h3>
8.
9.     <table>
10.         <tr>
11.             <td>NIM</td>
12.             <td>NAMA</td>
13.             <td>JURUSAN</td>
14.             <td>ALAMAT</td>
15.         </tr>
```

```
16.         @foreach($rows as $row)
17.         <tr>
18.             <td>{{ $row->mhsw_nim }}</td>
19.             <td>{{ $row->mhsw_nama }}</td>
20.             <td>{{ $row->mhsw_jurusan }}</td>
21.             <td>{{ $row->mhsw_alamat }}</td>
22.         </tr>
23.         @endforeach
24.     </table>
25. </div>
26.
27. @endsection
```

6. Berikutnya kita buat *Route* di **routes/web.php**, tambahkan baris perintah berikut ini :

```
Route::get('/mahasiswa', 'MahasiswaController@index');
```

7. Tambahkan link pada tempat yang anda inginkan, contoh pada *file resources/views/layouts/app.blade.php* tambahkan dibawah baris 58 seperti perintah berikut ini :

```
<a class="dropdown-item" href="{{ url('/mahasiswa') }}">Mahasiswa</a>
```



8. Simpan semua *file* praktikum anda dan coba akses kembali ke <http://localhost/oop-praktikum-10/public> pastikan semua berjalan lancar. Jika template css tidak di load, ubah saja pada layouts/app.blade.php pada baris 20

```
<link href="{{ asset('css/app.css') }}" rel="stylesheet">
```

Ganti dengan

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
bootstrap/4.5.0/css/bootstrap.min.css" crossorigin="anonymous">
```


Lihat di <https://getbootstrap.com/docs/4.5/getting-started/introduction/> untuk penggantian link. Materi Bootstrap akan dibahas pada Modul tersendiri.

The first screenshot shows a dashboard with a 'Dashboard' link and a 'You are logged in!' message. The second screenshot shows the same dashboard but with a 'Mahasiswa' link and a 'Logout' button added to the right sidebar. The third screenshot shows a 'Daftar Mahasiswa' form with fields for 'NIM', 'NAMA', 'JURUSAN', and 'ALAMAT'.

10.5 Latihan View

Ikuti langkah-langkah instalasi tersebut diatas dan praktikan pada komputer masing-masing. Ulangi langkah tersebut beberapa kali agar mudah untuk diingat.

10.6 Praktikum Create

Pada praktikum 11 sebelumnya kita sudah berhasil membuat link dan menu menggunakan konsep Model View Controller yang ditawarkan oleh framework Laravel. Sekarang kita lanjutkan dengan pembuatan form input data melanjutkan praktikum sebelumnya.

1. Buka kembali *file* **app/Http/Controllers/MahasiswaController.php** pastikan baris perintah berikut sudah ada.

```
1. <?php
2.
3. namespace App\Http\Controllers;
4.
5. use Illuminate\Http\Request;
6. use App\Mahasiswa;
7.
8. class MahasiswaController extends Controller
```

```
9. {
10.
11.     public function index()
12.     {
13.         $rows = Mahasiswa::all();
14.         return view('mahasiswa.index', compact('rows'));
15.     }
16.
17.     public function create()
18.     {
19.         return view('mahasiswa.add');
20.     }
}
```

Method **create()** akan mengarahkan browser ke halaman *form input* yang akan kita buat.

2. Berikutnya kita buat *Route* di **routes/web.php**, tambahkan baris perintah berikut ini :

```
Route::get('/mahasiswa/create', 'MahasiswaController@create');
Route::post('/mahasiswa', 'MahasiswaController@store');
```

3. Edit *file resources/views/mahasiswa/add.blade.php* dan tambahkan perintah berikut ini.

```
1. @extends('layouts.app')
2.
3. @section('content')
4.
5. <div class="container">
6.
7.     <h3>Tambah Data Mahasiswa</h3>
8.     <form action="{{ url('/mahasiswa') }}">
9.         @csrf
10.         <table>
11.             <tr>
12.                 <td>NIM</td>
13.                 <td><input type="text" name="mhs_w_nim"></td>
14.             </tr>
15.             <tr>
16.                 <td>NAMA</td>
17.                 <td><input type="text" name="mhs_w_nama"></td>
18.             </tr>
19.             <tr>
20.                 <td>JURUSAN</td>
21.                 <td><input type="text" name="mhs_w_jurusan"></td>
22.             </tr>
23.             <tr>
24.                 <td>ALAMAT</td>
25.                 <td><input type="text" name="mhs_w_alamat"></td>
26.             </tr>
27.             <tr>
28.                 <td></td>
```

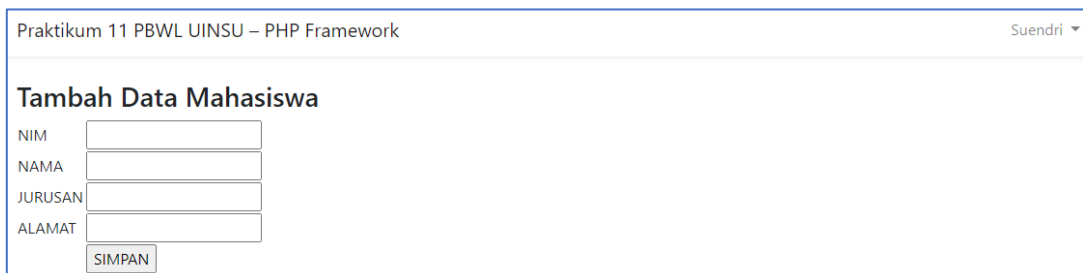
```
29.         <td><input type="submit" value="SIMPAN"></td>
30.     </tr>
31. </table>
32. </form>
33. </div>
34.
35. @endsection
```

4. Tambahkan *method* untuk menyimpan data pada file **app/Http/Controllers/MahasiswaController.php** seperti pada baris perintah berikut ini.

```
1. public function store(Request $request)
2. {
3.     $request->validate([
4.         'mhs_nim' => 'bail|required|unique:tb_mhs',
5.         'mhs_nama' => 'required'
6.     ],
7.     [
8.         'mhs_nim.required' => 'NIM wajib diisi',
9.         'mhs_nim.unique' => 'Nama Tahun sudah ada',
10.        'mhs_nama.required' => 'Nama wajib diisi'
11.    ]);
12.
13.    Mahasiswa::create([
14.        'mhs_nim' => $request->mhs_nim,
15.        'mhs_nama' => $request->mhs_nama,
16.        'mhs_jurusan' => $request->mhs_jurusan,
17.        'mhs_alamat' => $request->mhs_alamat
18.    ]);
19.
20.    return redirect('mahasiswa');
21. }
```

5. Silakan akses ke <http://localhost/oop-praktikum-10/public/mahasiswa/create> atau anda juga bisa membuat sebuah link pada view **index.blade.php** untuk diarahkan pada link tersebut dengan perintah:

```
<a href="{{ url('mahasiswa/create') }}">Tambah Data</a>
```



Praktikum 11 PBWL UINSU – PHP Framework

Tambah Data Mahasiswa

NIM

NAMA

JURUSAN

ALAMAT

Praktikum 11 PBWL UINSU – PHP Framework				Suendri ▾
Daftar Mahasiswa Tambah Data				
NIM	NAMA	JURUSAN	ALAMAT	
17021000	Faiz El Muhammadi	Sistem Informasi Medan		
17021001	Fauziah Fitri El Muhammadi	Sistem Informasi Medan		

10.6 Latihan Create

Ikuti langkah-langkah instalasi tersebut diatas dan praktikan pada komputer masing-masing. Ulangi langkah tersebut beberapa kali agar mudah untuk diingat.

10.7 Praktikum Edit

Pada modul praktikum 12 kita sudah berhasil membuat proses input data menggunakan framework Laravel. Modul praktikum 13 merupakan lanjutan proses berikutnya yaitu edit data. Edit data digunakan untuk melakukan pengubahan data jika terjadi kesalahan input. Ikuti langkah-langkah berikut untuk membuat proses edit data, Framework yang kita gunakan pada praktikum ini masih **oop-praktikum-10** agar tidak terpisah masing-masing prosesnya.

1. Buka kembali file **app/Http/Controllers/MahasiswaController.php** tambahkan baris perintah berikut ini atau edit nama method yang disediakan.

```
1. public function edit($id)
2. {
3.     $row = Mahasiswa::findOrFail($id);
4.     return view('mahasiswa.edit', compact('row'));
5. }
6.
7. public function update(Request $request, $id)
8. {
9.     $request->validate([
10.         'mhs_w_nim' => 'bail|required|unique:tb_mhs_w',
11.         'mhs_w_nama' => 'required'
12.     ],
13.     [
14.         'mhs_w_nim.required' => 'NIM wajib diisi',
15.         'mhs_w_nim.unique' => 'Nama Tahun sudah ada',
16.         'mhs_w_nama.required' => 'Nama wajib diisi'
17.     ]);
18.
19.     $row = Mahasiswa::findOrFail($id);
20.     $row->update([
21.         'mhs_w_nim' => $request->mhs_w_nim,
22.         'mhs_w_nama' => $request->mhs_w_nama,
23.         'mhs_w_jurusan' => $request->mhs_w_jurusan,
24.         'mhs_w_alamat' => $request->mhs_w_alamat
```

```
25.     });
26.
27.     return redirect('mahasiswa');
28. }
```

2. Berikutnya kita buat *Route* di **routes/web.php**, tambahkan baris perintah berikut ini :

```
Route::get('mahasiswa/{id}/edit', 'MahasiswaController@edit');
Route::patch('mahasiswa/{id}', 'MahasiswaController@update');
```

3. Edit *file resources/views/mahasiswa/edit.blade.php* dan tambahkan perintah berikut ini.

```
1. @extends('layouts.app')
2.
3. @section('content')
4.
5. <div class="container">
6.
7.     <h3>Edit Data Mahasiswa</h3>
8.     <form action="{{ url('/mahasiswa/' . $row->mhsw_id) }}" method="POST">
9.         <input name="_method" type="hidden" value="PATCH">
10.         @csrf
11.         <table>
12.             <tr>
13.                 <td>NIM</td>
14.                 <td><input type="text" name="mhsw_nim" value="{{ $row->mhsw_nim }}"></td>
15.             </tr>
16.             <tr>
17.                 <td>NAMA</td>
18.                 <td><input type="text" name="mhsw_nama" value="{{ $row->mhsw_nama }}"></td>
19.             </tr>
20.             <tr>
21.                 <td>JURUSAN</td>
22.                 <td><input type="text" name="mhsw_jurusan" value="{{ $row->mhsw_jurusan }}"></td>
23.             </tr>
24.             <tr>
25.                 <td>ALAMAT</td>
26.                 <td><input type="text" name="mhsw_alamat" value="{{ $row->mhsw_alamat }}"></td>
27.             </tr>
28.             <tr>
29.                 <td></td>
30.                 <td><input type="submit" value="UPDATE"></td>
31.             </tr>
32.         </table>
33.     </form>
34. </div>
35.
36. @endsection
```

4. Edit file **resources/views/mahasiswa/index.blade.php** dan tambahkan link edit sesuai yang diinginkan atau seperti pada contoh berikut ini:

```
1. @extends('layouts.app')
2.
3. @section('content')
4.
5. <div class="container">
6.
7.     <h3>Daftar Mahasiswa <a href="{{ url('mahasiswa/create') }}">Tambah Data</a>
8. </h3>
9.
10.    <table>
11.        <tr>
12.            <td>NIM</td>
13.            <td>NAMA</td>
14.            <td>JURUSAN</td>
15.            <td>ALAMAT</td>
16.            <td>EDIT</td>
17.        </tr>
18.        @foreach($rows as $row)
19.            <tr>
20.                <td>{{ $row->mhsw_nim }}</td>
21.                <td>{{ $row->mhsw_nama }}</td>
22.                <td>{{ $row->mhsw_jurusan }}</td>
23.                <td>{{ $row->mhsw_alamat }}</td>
24.                <td><a href="{{ url('mahasiswa/' . $row->mhsw_id . '/edit') }}">Edit</a></td>
25.            </tr>
26.        @endforeach
27.    </table>
28. </div>
29. @endsection
```

6. Silakan akses ke <http://localhost/oop-praktikum-10/public/mahasiswa> klik link salah satu untuk melakukan perubahan pada baris tersebut.

Praktikum 11 PBWL UINSU – PHP Framework				Login	Register
Daftar Mahasiswa Tambah Data					
NIM	NAMA	JURUSAN	ALAMAT	EDIT	
17021000	Faiz El Muhammadi	Sistem Informasi Medan		Edit	
17021001	Fauziah Fitri El Muhammadi	Sistem Informasi Medan		Edit	
17021002	Faiz Rizky	Sistem Informasi Medan		Edit	

Praktikum 11 PBWL UINSU – PHP Framework Login Register

Edit Data Mahasiswa

NIM	<input type="text" value="17021002"/>
NAMA	<input type="text" value="Faiz Rizky"/>
JURUSAN	<input type="text" value="Sistem Informasi"/>
ALAMAT	<input type="text" value="Medan"/>
<input type="button" value="UPDATE"/>	

5. Selanjutnya kita akan buat link untuk Delete, sesuai dengan namanya link ini akan kita gunakan untuk menghapus data jika tidak diperlukan. Buka kembali *file* **app/Http/Controllers/MahasiswaController.php** tambahkan baris perintah berikut ini atau edit nama *method* yang disediakan.

```
1. public function destroy($id)
2. {
3.     $row = Mahasiswa::findOrFail($id);
4.     $row->delete();
5.
6.     return redirect('mahasiswa');
7. }
```

6. Berikutnya kita buat *Route* di **routes/web.php**, tambahkan baris perintah berikut ini :

```
Route::delete('mahasiswa/{id}', 'MahasiswaController@destroy');
```

7. Edit file **resources/views/mahasiswa/index.blade.php** dan tambahkan link edit sesuai yang diinginkan atau seperti pada contoh berikut ini:

```
1. <form action="{ url('mahasiswa/' . $row->mhs_id) }}" method="POST">
2. <input name="_method" type="hidden" value="DELETE">
3. @csrf
4. <button>Hapus</button>
5. </form>
```

8. Silakan akses ke <http://localhost/oop-praktikum-10/public/mahasiswa> klik link salah satu untuk menghapus baris tersebut.

Praktikum 11 PBWL UINSU – PHP Framework					Login Register	
Daftar Mahasiswa Tambah Data						
NIM	NAMA	JURUSAN	ALAMAT	EDIT	HAPUS	
17021000	Faiz El Muhammadi	Sistem Informasi	Medan	Edit	<input type="button" value="Hapus"/>	
17021001	Fauziah Fitri El Muhammadi	Sistem Informasi	Medan	Edit	<input type="button" value="Hapus"/>	
17021002	Faiz Rizky	Sistem Informasi	Medan	Edit	<input type="button" value="Hapus"/>	

10.8 Latihan Edit

Ikuti langkah-langkah instalasi tersebut diatas dan praktikan pada komputer masing-masing. Ulangi langkah tersebut beberapa kali agar mudah untuk diingat.

DAFTAR PUSTAKA

- Sidik, Betha. 2005. *Pemrograman web dengan HTML*. Bandung: Informatika Bandung.
- Jubile Enterprise, 2012. *Buku Pintar HTML5+CSS3+Dreamweaver CS6*. Jakarta: PT Elex Media Komputindo.
- Sakur, Stendy B. 2010. *PHP5 Pemrograman Berorientasi Objek Konsep dan Implementasi*. Yogyakarta: Penerbit Andi.
- Solichin Achmad. 2010. *MySQL 5 Dari Pemula Hingga Mahir*. Tersedia di <http://achmatim.net>
- Solichin Achmad. 2010. *Pemrograman Web dengan PHP dan MySQL*. Tersedia di <http://achmatim.net>
- Yuana, Rosihan Ari. 2012. *Panduan Praktis OOP di PHP*. Tersedia di <http://blog.rosihanari.net>